НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено
Завідувач кафедри
_____ *С.Г. Стіренко*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки **6.050102 «Комп’ютерна інженерія»**

на тему «Система доповненої реальності для мобільної платформи iOS»

Виконав: студент *4* курсу, групи *ІО-53*

Філіпенко Роман Олексійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *старший викладач Алещенко О.В.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант основна частина *старший викладач Алещенко О.В.* _____
(назва розділу) (посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант нормоконтроль *д.т.н, проф. Сімоненко В.П* _____
(назва розділу) (посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Анотація

В бакалаврській роботі розглядається система доповненої реалності та її реалізація на мобільній операційній системі iOS. Як практична сторона розроблена программа для виміру відстаней та 3D візуалізації предметів за допомогою технології доповненої реальності.

Програмний продукт був створений на мові Swift у середовищі розробки Xcode 10.2.1.

Annotation

This bachelor thesis examines the system of augmented reality and its implementation on the mobile operating system iOS. As a practical side, a program for measuring distances and 3D visualization of objects with the help of augmented reality technology has been developed.

The software product was created in Swift language in the development environment Xcode 10.2.1.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ» ім. ІГОРЯ СІКОРСЬКОГО

Кафедра обчислювальної техніки

Напрям підготовки - 6.050102 - «Комп'ютерна інженерія»

Затверджую:

зав. кафедрою

_____ *С.Г. Стіренко*
(підпис) (ініціали, прізвище)

«____»_____2019

ЗАВДАННЯ

на бакалаврську дипломну роботу студента

Філіпенка Романа Олексійовича

1. Тема проекту (роботи) *Система доповненої реальності для мобільної платформи iOS*

керівник проекту (роботи) *старший викладач Алещенко О.В.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджена наказом по університету від «__» _____ 20*19*р. №_____

2. Термін здачі студентом закінченого роботи_____*2019р.*
3. Вихідні дані до роботи *технічне завдання, теоретичні данні.*
4. Зміст розрахунково-пояснювальної записки: *опис предметної області, дослідження методів створення систем доповненої реальності, програма для знаходження розмірів предметів та візулізація предметів у 3D над маркерами за допомогою технології доповненої реальності.*

5. Консультанта роботи, з вказівкою розділів роботи, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Симоненко В. П.		

6. Дата видачі завдання *15.12.2018 року*

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту (роботи)	Строк виконання етапів проекту(роботи)	Примітки
1.	*Затвердження теми роботи*	*15.12.2018*	
2.	*Вивчення та аналіз завдання*	*30.03.2019*	
3.	*Розробка архітектури та загальної структури систем*	*10.04.2019*	
4.	*Розробка структур окремих підсистем*	*22.04.2019*	
5.	*Програмна реалізація системи*	*28.04.2019*	
6.	*Оформлення пояснювальної записки*	*04.05.2019*	
7.	*Передзахист*	*28.05.2019*	
8.	*Захист*	*19.06.2019*	

Студент-дипломник _____
(підпис)

Керівник роботи _____
(підпис)

Опис альбому
до дипломного проекту

на тему: «Система доповненої реальності для мобільної платформи iOS»

Київ – 2019

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	ІАЛЦ.467100.001 ВП	Відомість проекту	1	
3	А4	ІАЛЦ.467100.002 ТЗ	Технічне завдання	3	
4	А4	ІАЛЦ.467100.003 ПЗ	Пояснювальна записка	65	
5	А4	ІАЛЦ.467100.004 Д1	Схема функціональна Блок-схема алгоритму	1	
6	А4	ІАЛЦ.467100.005 Д2	Схема структурна Діаграма класів	1	
7	А4	ІАЛЦ.467100.006 Д3	Схема взаємодії компонентів програми	1	
8	А4	ІАЛЦ.467100.007 Д4	Лістинг програми	4	

					ІАЛЦ.467100.001 ВП	
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розроб.		*Філіпенко Р.О.*			*Відомість дипломного проекту*	*Літ.* / *Арк.* / *Акрушів*
Перевір.		*Алещенко О.В.*				*1* / *1*
Н. Контр.		*Симоненко В.П.*				*НТУУ «КПІ», ФІОТ ІО-53*
Затверд.						

Технічне завдання
до дипломного проекту

на тему: «Система доповненої реальності для мобільної платформи iOS»

Київ – 2019

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Назва розробки: Система доповненої реальності для мобільної платформи iOS.

Галузь застосування: Додаток, яким зможуть користуватися як для розваг, так і для роботи.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування, затверджене кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут» ім. Ігоря Сікорського.

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка системи доповненої реальності для мобільної платформи iOS для набуття навичок та можливості доопрацювати проект у майбутньому.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література з теорії і практики програмування, бакалаврські роботи інших студентів, публікації в Інтернеті з даних питань.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробляємого продукту

- Простий і інтуїтивно-зрозумілий інтерфейс системи.
- Можливість точно вимірювати відстані у навколишньому середовищі.
- Можливість точно відображати 3D-модель на роздрукованих картках при будь-якому куті нахилу.
- Технічна коректність отримуємих вимірів.

5.2. Вимоги до програмного забезпечення

- Операційна система iOS 11.0 та вище.

5.3. Вимоги до апаратної частини

- iPhone 6S та вище.
- iPad Air 2 та вище.

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення літератури	20.03.2019
Складання і узгодження технічного завдання	01.04.2019
Створення модулів системи, що розробляється	10.04.2019
Тестування окремих модулів системи	20.04.2019
Допрацювання, налагодження і виправлення помилок	01.05.2019
Оформлення документації дипломної роботи	10.05.2019

Пояснювальна записка
до дипломного проекту

на тему: «Система доповненої реальності для мобільної платформи iOS»

Київ – 2019

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AR (Augmented Reality) Доповнена реальність

VR (Virtual Reality) Віртуальна реальність

MR (Mixed Reality) Змішана реальність

GPS (Global Positioning System) Система глобального позиціонування

KARMA (Knowledge-based Augmented Reality for Maintenance Assistance) Інтерактивний помічник з обслуговування

QR-code (Quick Response code) Код швидкої відповіді

WWDC (Worldwide Developers Conference) Всесвітня конференція розробників

HUD (Head-Up Display) Прозорий дисплей

HMD (Head-Mounted Display) Головний дисплей

ГЛОНАСС Глобальна Навігаційна Супутникова Система

2D (2-dimensional) Двовимірний простір

3D (3-dimensional) Тривимірний простір

SLAM (Simultaneous Localization And Mapping) Одночасна локалізація і картографування

EKF SLAM (Extended Kalman Filter for Simultaneous Localization And Mapping) Розширений фільтр Калмана для одночасної локалізація і картографування

RGB (Red, Green, Blue) Червоний, зелений, синій

IMU (Inertial Measurement Unit) Інерційний вимірювальний пристрій

FPS (Frames Per Second) Кількість кадрів в секунду

FoV (Field of View) Поле зору

API (Application Programming Interface) Прикладний програмний інтерфейс

SDK (Software Development Kit) Набір із засобів розробки

VIO (Visual Inertial Odometry) Візуальна інерційна одометрія

6DOF (6 Degrees Of Freedom) Шість ступенів свободи

ВСТУП

Доповнена реальність - це середовище, яке в реальному часі доповнює фізичний світ, яким ми його бачимо, цифровими даними за допомогою будь-яких пристроїв - планшетів, смартфонів або інших, і програмної частини. Наприклад, Google Glass або шолом Залізної Людини. Системи прицілювання в сучасних бойових літаках - це теж доповнена реальність.

Доповнену реальність (augmented reality, AR) треба відрізняти від віртуальної (virtual reality, VR) і змішаної (mixed reality, MR). У доповненій реальності віртуальні об'єкти проектуються на реальне оточення.

Віртуальна реальність - це створений технічними засобами світ, який передається людині через органи чуття. Змішана або гібридна реальність об'єднує обидва підходи. Тобто, віртуальна реальність створює свій світ, куди може зануритися людина, а доповнена додає віртуальні елементи в світ реальний. Виходить, що VR взаємодіє лише з користувачами, а AR - з усім зовнішнім світом.

У користувача AR інформація відображається на екрані окулярів або контактних лінз. Інформація про те, що бачить користувач, з’являється над потрібним об’єктом. Користувач може дивитися на предмети як зазвичай, але перед його очима будуть з'являтися підказки про місце, в якому він перебуває, предмети, які бачить, та історичні довідки про події, що сталися в цьому місці.

Наприклад, замість того, щоб читати дрібний текст на упаковці товару, користувач може побачити інформацію про його склад та дату виготовлення у тій частині поля свого зору, в якій він звик бачити подібну інформацію. Або користувачу не потрібно буде дивитися у дорожню карту чи користуватися окремим GPS-навігатором, тому що необхідний напрям буде вказано стрілкою прямо перед його очима. Тож доповнена реальність має великий потенціал для розвитку комп'ютерних додатків.

Варто відзначити, що в даний час система освіти побудована таким чином, що учні отримують більше теоретичних знань, ніж практичних

навичок. Однак не можна заперечувати той факт, що теоретичні знання, застосовані на практиці, запам'ятовуються краще.

Складність проведення більшої кількості практичних занять викликана тим, що існують ризики пошкодження дорогого обладнання або нанесення шкоди здоров'ю учнів. Ще однією проблемою сучасної освіти є відсутність візуалізації деяких досліджуваних об'єктів. Наприклад, викладачі, розповідаючи про рідкісні породи мінералів, не мають можливості продемонструвати їх учням. І тут на допомогу може прийти доповнена реальність.

У якості цілі бакалаврського дипломного проекту було взято вивчення технології доповненої реальності та знаходження методів використання цієї технології на мобільних засобах. У данному випадку – на мобільній операційній системі iOS.

Тож для досягнення поставлених цілей необхідно вирішити наступні задачі:

1. Дослідити поняття доповненої реальності та історію його виникнення.
2. Розглянути сфери, де можна застосувати технологію доповненої реальності.
3. Виділити найбільш актуальну тему і можливість її реалізації на мобільній платформі.
4. Розробити та створити мобільний додаток для операційної системи iOS з використанням технології доповненої реальності.

РОЗДІЛ 1.

1. ОГЛЯД ТЕХНОЛОГІЇ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

1.1. Опис предмету вивчення

Доповнена реальність є одним з перспективних напрямків сучасних IT-розробок. Дану технологію можна назвати новим способом отримання доступу до даних.

Доповнена реальність - це середовище з накладенням віртуальної інформації, такої як текст, графіка, аудіо, на елементи реального життя в режимі реального часу. Доповнену реальність не варто плутати з віртуальною реальністю: віртуальна реальність - це віртуальний світ, створений за допомогою технічних засобів, що передається людині за допомогою органів дотику, нюху, зору і слуху, а доповнена реальність має на увазі додаток реального світу віртуальними об'єктами. Також існує термін доповненої віртуальності, який має на увазі віртуальну реальність, доповнену об'єктами реального світу.

При створенні доповненої реальності в простір в режимі реального часу поміщаються об'єкти за допомогою спеціального програмного забезпечення і гаджетів, таких як, наприклад, окуляри доповненої реальності, планшети, смартфони з функцією AR і інші гаджети. Також варто зважати на те, що деякі технології доповненої реальності потребують наявність спеціальної мітки, за допомогою якої програма зможе помістити віртуальний обьект у реальний світ. Проте на сьогоднішній день потужності мобільних ґаджетів достатньо для позиціонування віртуальних предметів у реальному світі без спецільних міток.

1.2. Історія створення та розвитку технології доповненої реальності

Як багато інших цікавих дослідження, історія маніпуляцій з реальністю починається в науковій фантастиці. Автор «Чарівника країни Оз» Лайман Френк Баум в романі «Головний ключ» описав якийсь пристрій, здатне

позначати в режимі реального часу людей буквами, що вказують на їх характер і рівень інтелекту. Примітивні інструменти доповнення реальності були відомі задовго до того: це і маски, які одягали римські лучники, щоб краще цілитися, і підзорні труби з нанесеними мітками відстаней і так далі.

Але історія доповненої реальності, якою ми її знаємо зараз, бере початок з розробок, що стосуються Виаро. Батьком віртуальної реальності вважається Мортон Хейліг. Він отримав це звання за дослідження і винаходи, зроблені в 1950-х і 60-х роках. 28 серпня 1962 він запатентував симулятор Sensorama, зображений на рис. 1.1. Сам Хейліг ще називав його театром занурення.

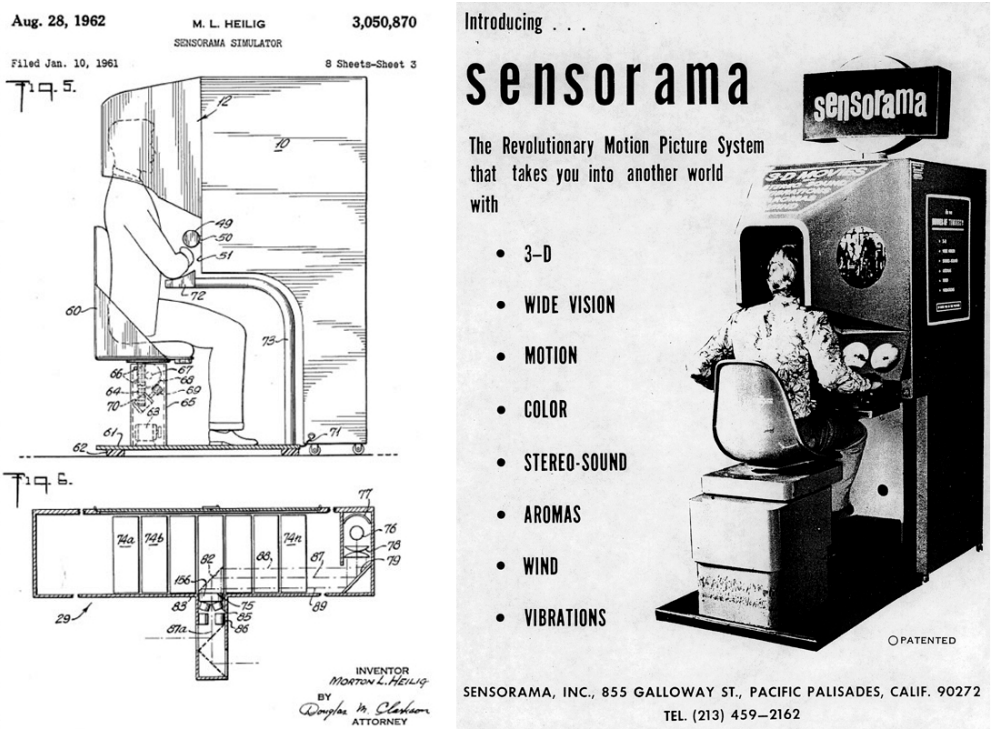

Рис. 1.1. Опис патенту та рекламна брошура симулятору «Sensorama»

Користувач міг зробити віртуальну поїздку на мотоциклі по вулицях Брукліна. Ефект присутності досягався шляхом впливу на всі основні органи чуття одночасно: екран демонстрував запис «від першої особи», зняту одночасно трьома кінокамерами, сидіння тремтіло, вентилятори створювали відчуття зустрічного вітру, стереодинаміки транслювали звуки жвавої вулиці, в камеру надходили відповідні запахи.

Патент описує віртуальну технологію, в якій візуальні образи доповнюються рухами повітря і вібрацією. Обгрунтування її існування

давалося таке: «Сьогодні постійно зростає попит на методи навчання і тренування людей таким способом, щоб виключити ризики і небезпеку реальних ситуацій».

Це був пристрій ранньої версії віртуальної реальності, а не доповненої, але саме воно дало поштовх до розвитку обох напрямків. Хейліг навіть винайшов спеціальну 3D-камеру, щоб знімати фільми для Sensorama.

А ось в 1968-му році комп'ютерний фахівець і професор Гарварда Айван Сазерленд зі своїм студентом Бобом Спрауллом розробили пристрій, що одержав назву «Дамоклів меч», зображений на рис. 1.2. І це була перша система вже саме доповненої реальності на основі головного дисплея.

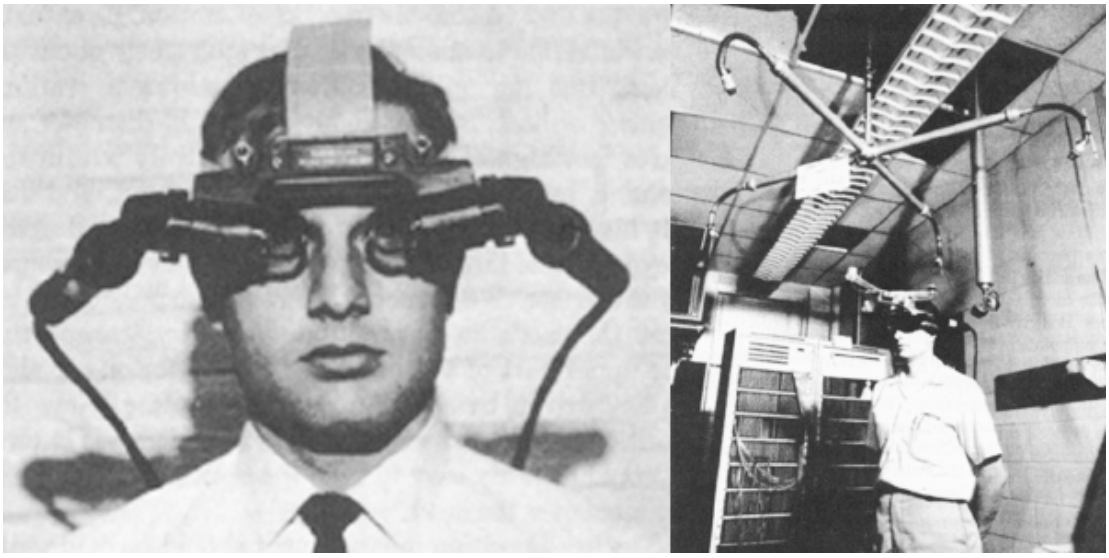

Рис. 1.2. Використання пристрою «Дамоклів меч»

Окуляри були настільки важкими, що їх довелося кріпити до стелі. Конструкція загрозливо нависала над випробуваним, звідси і назва. В окуляри зі стереоскопічним дисплеєм транслювалася проста картинка з комп'ютера. Перспектива спостереження за об'єктами змінювалася залежно від рухів голови користувача, тому знадобився механізм, що дозволяє відслідковувати напрямок погляду. Для того часу це був фантастичний прорив.

Наступним кроком було створення Майроном Крюгером лабораторії зі штучною реальністю Videoplace в 1974-му році. Його основною метою було позбавити користувачів від необхідності надягати спеціальні шоломи, окуляри та інші пристосування для взаємодії зі штучною реальністю. У Videoplace

використовувалися проектори, відеокамери та інше обладнання. Люди, перебуваючи в різних кімнатах, могли взаємодіяти один з одним. Їх руху записувалися на відео, аналізувалися і переводилися в силуети штучної реальності. Користувачі бачили, як їхні силуети взаємодіють з об'єктами на екрані і це створювало враження, що вони частина штучної реальності. Хоча правильніше було б назвати це проектом інтерактивного оточення.

Перше масове використання доповненої реальності стало можливо завдяки Дену Рейтон, який в 1982-му році використовував радар і камери в космосі для того, щоб показати рух повітряних мас, циклонів і вітрів в телепрогнозах погоди. Там AR досі використовується таким чином.

У 90-ті пошук нових способів використання продовжився, а вчений Том Коделл вперше запропонував термін «доповнена реальність». Перед ним і його колегою поставили завдання: знизити витрати на дорогі діаграми, які використовували для розмітки заводських зон по збірці літаків Boeing. І рішенням стала заміна фанерних знаків з позначеннями на спеціальні шоломи, які відображали інформацію для інженерів. Це дозволило не переписувати позначення кожного разу вручну, а просто змінювати їх в комп'ютерній програмі. На рис. 1.3 зображено інтерфейс головного дисплею.

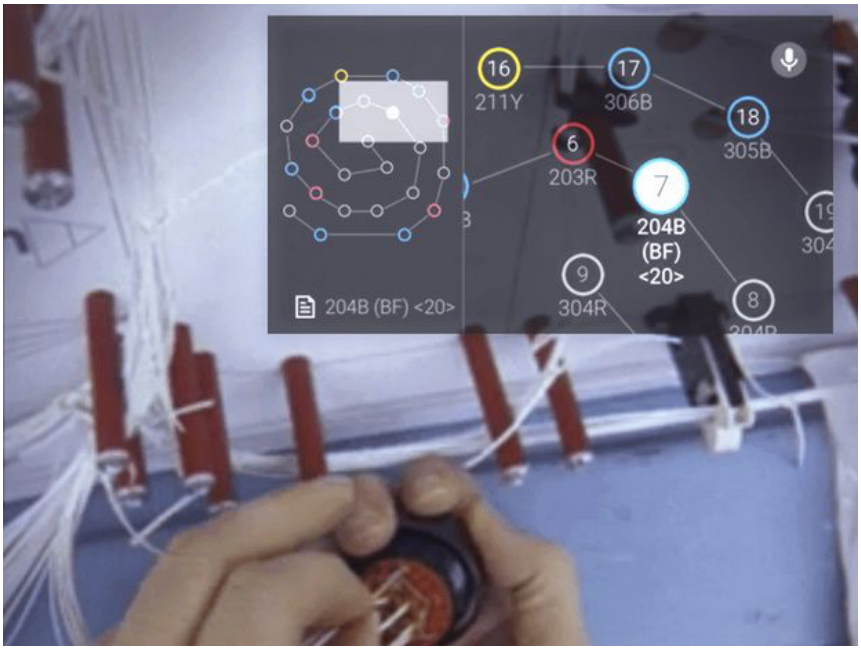

Рис. 1.3. Інтерфейс головного дисплею працівника компанії «Boeing»

Далі розвиток відбувався стрімко. Стрибок, зроблений у виробництві мікропроцесорів, і, як наслідок, у всьому технологічним секторі, дозволив сильно прискорити роботи.

У 1993-му році в університеті штату Колумбія Стів Файнер представив систему KARMA (Knowledge-based Augmented Reality for Maintenance Assistance, перекладається приблизно як «Інтерактивний помічник з обслуговування»), зображену на рис. 1.4, яка дозволяла через шолом віртуальної реальності побачити інтерактивну інструкцію з обслуговування принтера.

Рис. 1.4. Приклад використання системи « KARMA»

А ось в 95-м Джун Рекімото зібрав Navicam - прототип мобільного пристрою доповненої реальності, якою її зараз знають користувачі смартфонів. Navicam представляв собою портативний дисплей із закріпленою на зворотному боці камерою, чий відеопотік оброблявся комп'ютером і, при виявленні кольоровий мітки, виводив на екран інформацію про об’єкт, як зображено на рис. 1.5.

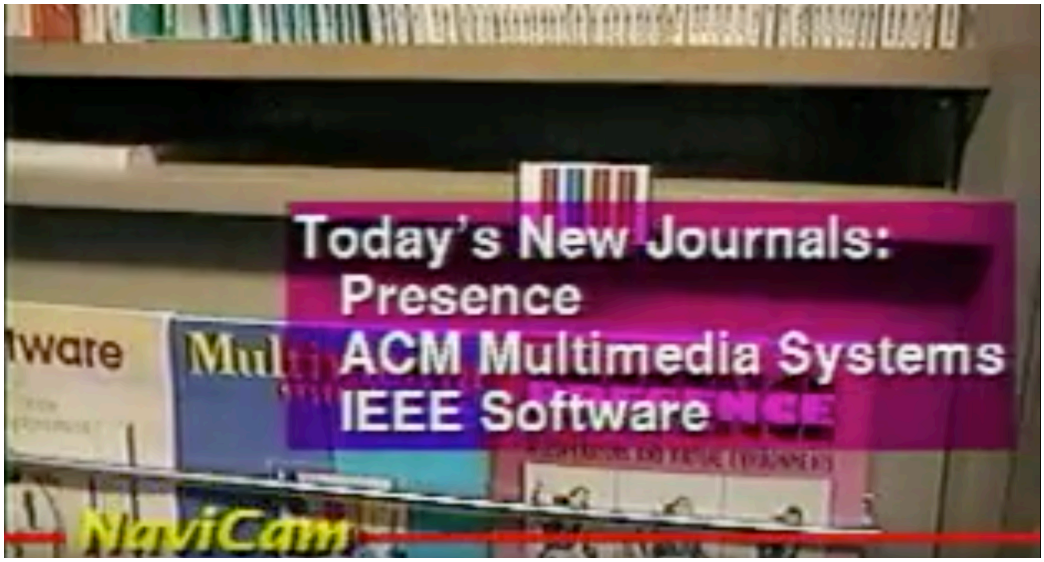

Рис. 1.5. Приклад розпізнавання мітки системою «Navicam»

У 96-му році Джуном Рекімото і Южді Аятцука був розроблений Матричний Метод (або КіберКод). Він описує реальні і віртуальні об'єкти за допомогою плоских міток на зразок QR-кодів. Це дозволяло вписувати віртуальні речі в реальний світ, просто переносячи мітки. Наприклад, покласти на підлогу листок з кодом, навести на кімнату камерою - і ось у вас в кімнаті стоїть автомобіль.

Дослідник Рональд Азума у 1997 році дав визначення доповненій реальності як системі, яка неодмінно містить у собі наступні можливості:

1. Поєднувати реальне та віртуальне
2. Взаємодія у реальному часі
3. Працює у 3D

У 1999 році Хироказу Като створив відкриту бібліотеку для створення програмного забезпечення з AR-функціоналом ARToolKit (ще на GitHub). У ній використовувалася система розпізнавання положення і орієнтації камери в реальному часі. Це дозволяло стикувати картинку реальну і віртуальну з камери, що давало можливість рівно накладати шар комп'ютерної графіки на маркери реального світу. Можна сказати, що з релізом першої версії цієї бібліотеки почався сучасний етап активного розвитку доповненої реальності.

1.3. Сфери застосування технології доповненої реальності

Технології доповненої реальності дають нам відчути абсолютно нові властивості об'єктів і отримати нові відчуття від звичних речей, використовуючи стандартний комп'ютер і стандартні периферійні пристрої. Отже, кількість сфер, де можна знайти застосування технології доповненої реальності обмежено тільки фантазією творців і розробників.

Завдяки дуже швидкому прогресу ми можемо з легкістю створювати программи з технологією віртуальної реальності, хоча ще декілька років тому це було під силу лише ентузіастам та великим компаніям.

Щоб максимально ефективно інтегрувати доповнену реальність в справжню реальність, необхідно дотримуватися ряду важливих умов, таких як:

1. Мінімальні витрати на створення додатків для широкого використання.

Для того щоб елементи технології доповненої реальності можна було застосовувати в популярних додатках, вартість розробки таких елементів потрібно зменшувати, залучаючи, тим самим більше число фахівців в цю область.

2. Правдоподібне розташування віртуальних об'єктів.

Віртуальний об'єкт, будучи розміщеним у встановленій точці реально спостерігається сцени, зобов'язаний так себе вести, щоб у людини формувалося враження, ніби цей об'єкт насправді є частиною реально спостерігається сцени. Візуальна інформація для комфортного сприйняття повинна оновлюватися не менше 15 разів на секунду, проте бажано не менше 30 разів, інакше зображення буде занадто переривчастим. Кожні затримки в позиціонуванні або реакції предмета будуть вкрай помітні, і будуть виділяти його на загальному тлі.

3. Віртуальні і реальні об'єкти візуально не повинні сильно відрізнятися.

На додаток до фотореалістичного зображення віртуальних предметів, що само по собі є невід'ємною вимогою до доповненої реальності, візуальне поєднання існуючих і неіснуючих об'єктів має відбуватися коректно. Це, по суті, є однією з найбільш важкореалізованих умов. Так як ми абсолютно не

знаємо звідки і куди дивиться людина, на які об'єкти і які геометричні характеристики вони мають. Отже, не уявляючи геометричних характеристик і відстаней між реальними об'єктами, вкрай складно однозначно і правильно розмістити в просторі віртуальні об'єкти щодо реальних. Варто сказати що в сфері віртуальної реальності це питання майже зовсім не стоїть, так як всі видимі об'єкти створюються програмою, якраз виходячи з їх характеристик один одного.

4. Віртуальні об'єкти зобов'язані підкорятися законам фізики реального світу.

В першу чергу це пов'язано з ситуаціями зіткнення віртуальних і реальних об'єктів. Людина може пересуватися як їй завгодно в доповненому просторі, доповнена реальність повинна забезпечувати можливість пересування в просторі без будь-яких механічних обмежень. Система доповненої реальності повинна бути нескладна в налаштуванні і запуску, отже повинна бути доступна широкому колу потенційних користувачів, що не володіють спеціальними знаннями і навичками. Для того, щоб визначити положення спостерігача велика кількість систем доповненої реальності вимагають певного калібрування камери, яка стежить за реальною сценою. Процес калібрування досить непростий, особливо для камер з фіксованою фокусною відстанню. Стандартні WEB-камери, як правило, не можуть змінювати фокусну відстань, а якщо і можуть, то тільки шляхом програмної обробки вже отриманого зображення. Особливо дана проблема стосується девайсів на операційній системі Android, так як дуже складно реалізувати калібрування настільки різноманітної маси пристроїв.

Ще в 1984-му році у фільмі «Термінатор» Джеймса Кемерона була візуалізувана концепція доповненої реальності і комп'ютерного зору. Але Кемерон сильно випередив час, тому що вбудувати AR прямо в око в ті роки було неможливо навіть у найсміливіших фантазіях. Ідеалом бачилися форм-фактори контактних лінз або окулярів. Перше і зараз лише на стадії концептів, а ось у міру здешевлення і появи більш тонких виробничих процесів форма

окулярів ставала все ближче. З роками до неї остаточно приєднався і другий варіант реалізації - за допомогою ставших всюдисущими смартфонів.

Найгучнішою подією доповненої реальності останніх років стали випущені в 2013-му році окуляри Google Glass, що зображені на рис. 1.6, з якими є невелика плутанина. Незважаючи на те, що саме вони багатьом першими приходять на думку, коли мова заходить про доповнену реальності, до такої ці окуляри відносини майже не мали. Віртуальне середовище практично не взаємодіяло з реальним. Хіба що навігацію можна зарахувати до AR-контенту, але і вона була реалізована в стилі карт для телефону, а не якихось висячих над дорогою стрілок.

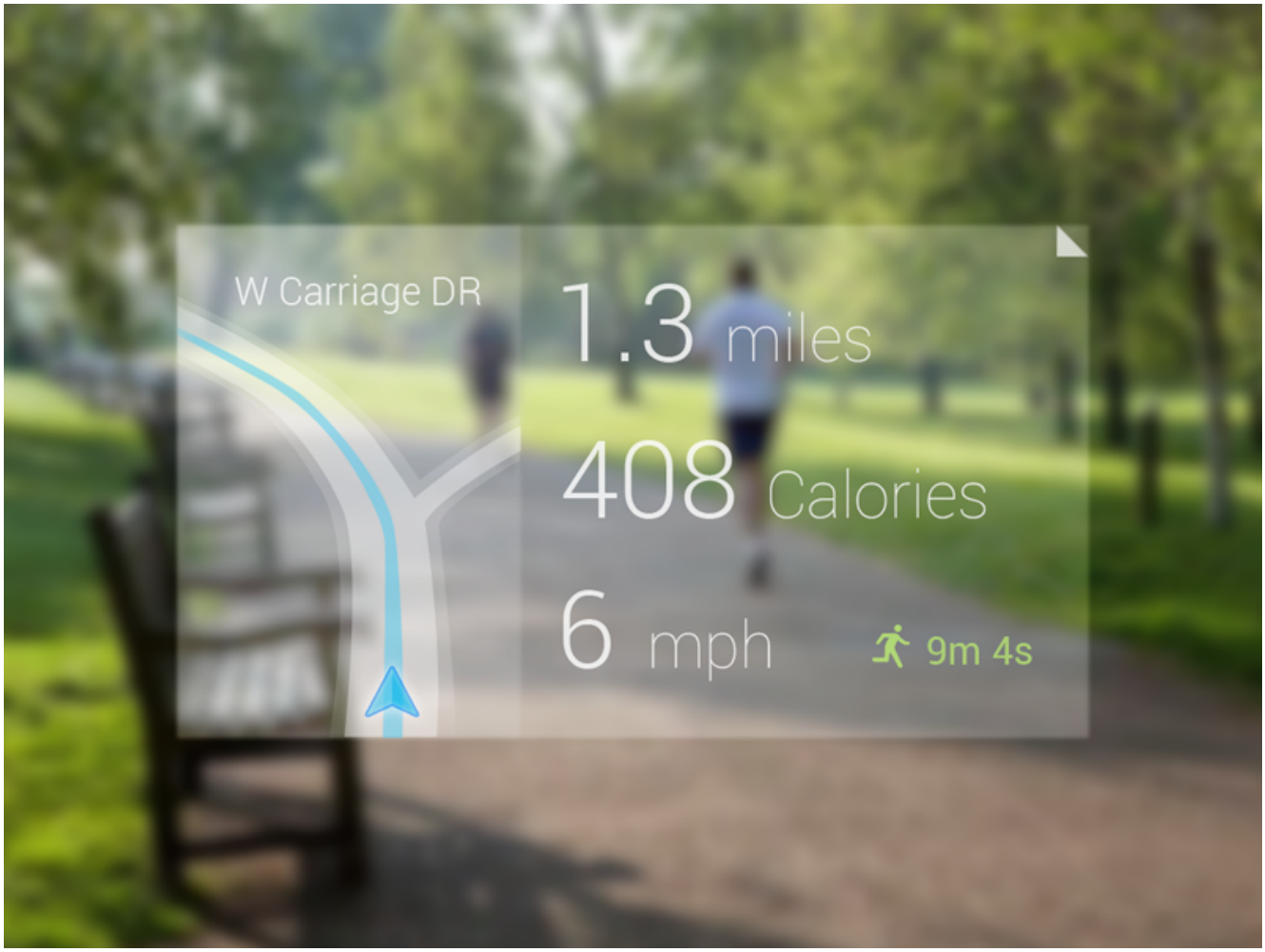

Рис. 1.6. Інтерфейс навігації у «Google Glass»

Через декулька років, а саме у 2016, компанія Microsoft презентувала своє бачення окулярів доповненої реальності Hololens. Hololens не вимагають підключення до іншого ПК або телефону. У окулярів чотири камери, за

допомогою яких вони аналізують кімнату і поєднують віртуальні об'єкти з реальним світом.

Окуляри дозволяють практично повноцінно працювати з Windows 10, причому, назва «Windows» набуває нового сенсу: вікна системи легко вішаються на стіни на манер, власне, вікон, як зображено на рис. 1.7. Окуляри запам'ятовують приміщення, тому, коли користувач повертається в ту ж саму кімнату, всі вікна додатків і інші елементи змішаної реальності чекають його на своїх місцях.

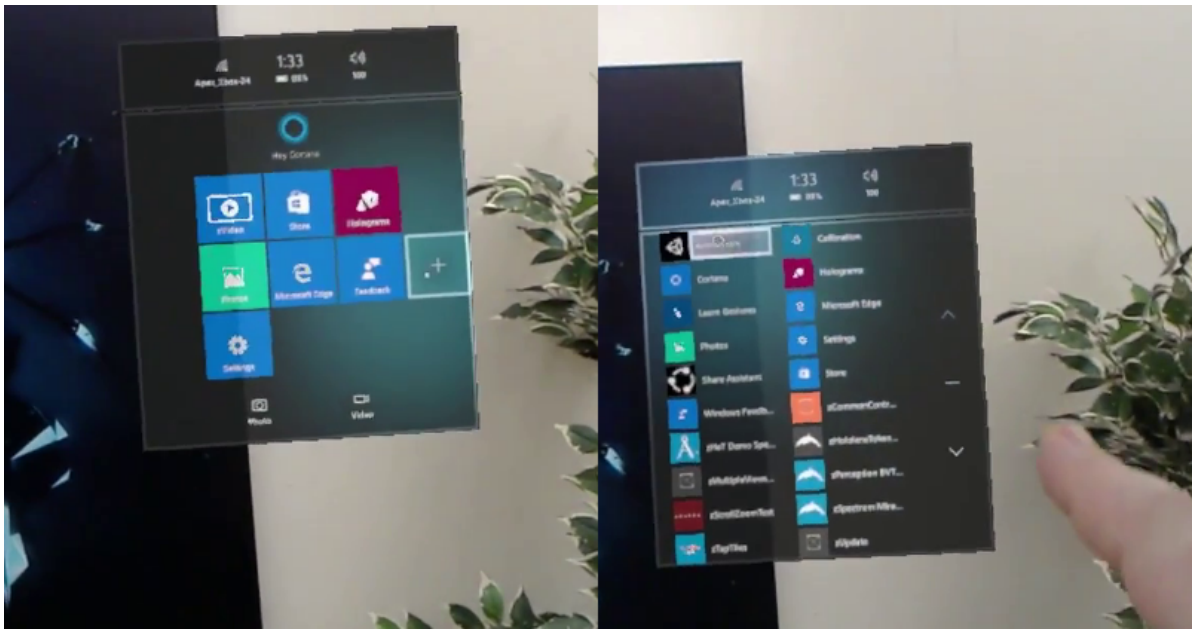

Рис. 1.7. Інтерфейс операційної системи «Windows» у окулярах доповненої реальності «Hololens»

У медицині ці технології доповненої реальності використовуються для створення реалістичних тренажерів, які дозволяють лікарям тренуватися і проводити різні хірургічні операції. При цьому інтерактивність і реалістичність тренажерів дозволять запобігти помилкам лікарів при проведенні справжніх операцій, як зображено на рис. 1.8. Або ж одразу бачити результати ЕКГ пацієнта, як на рис. 1.9.

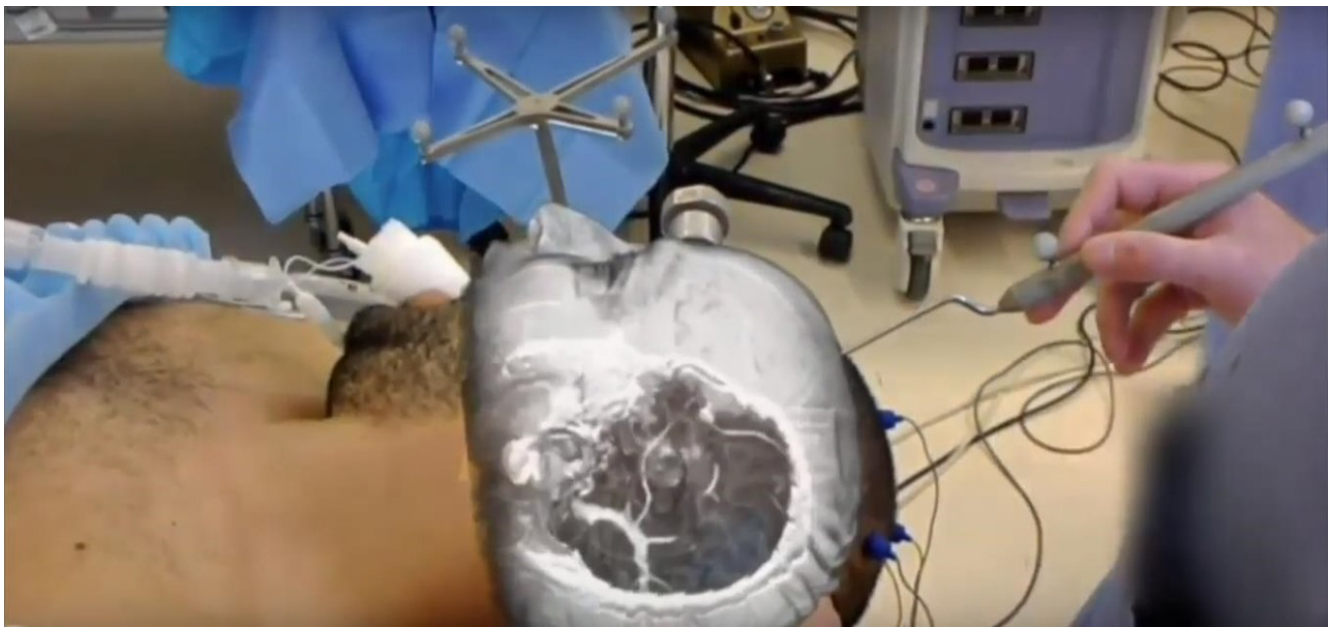

Рис. 1.8. Інтерфейс окулярів доповненої реальності «Hololens» для хірурга

Рис. 1.9. Інтерфейс ЕКГ у окулярах доповненої реальності «Hololens»

Технологія може зайняти ту нішу, яка в науковій фантастиці віддана голограмам. Тільки голограми будуть ще не скоро, а пристрої на кшталт Hololens технічно майже готові. Перспектива побачити в вузах, а після і школах, віртуальні інтерактивні ілюстрації, які можна розглянути з усіх боків як зображено на рис. 1.10., з якими можна взаємодіяти і тут же бачити результат своїх дослідів, є прекрасною фантазією про майбутнє, на порозі якого ми стоїмо. Навчання будь-яким інженерних спеціальностей може стати куди більш наочним і легким для розуміння.

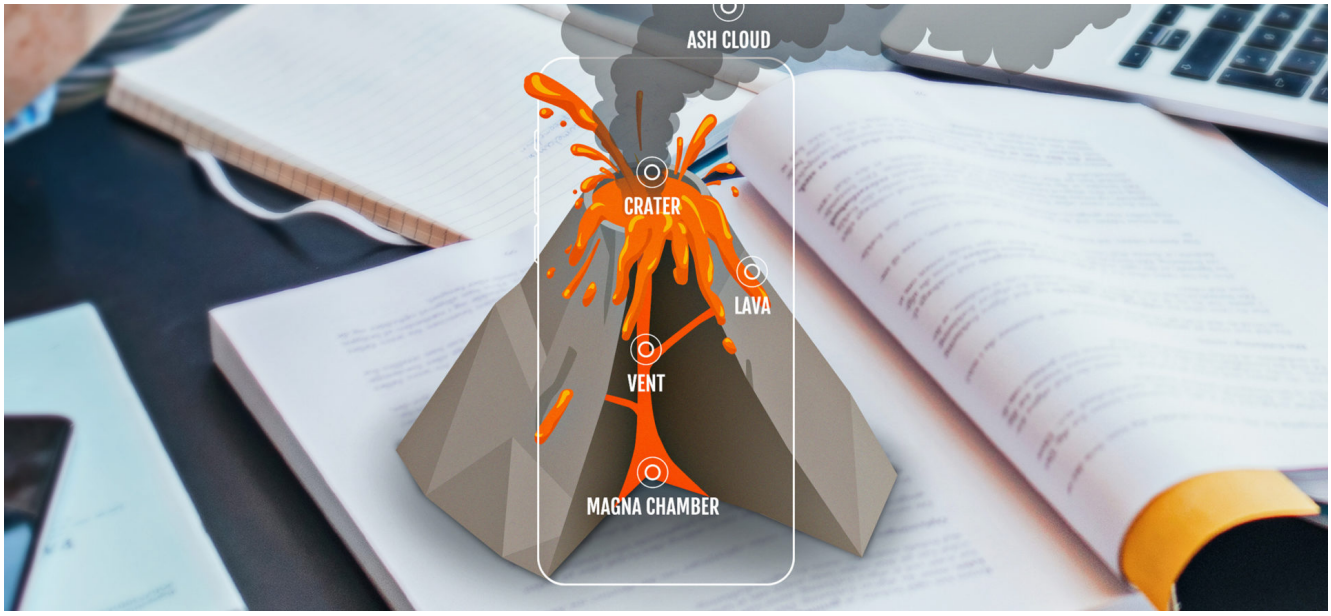

Рис. 1.10. Візулізація виверження вулкану у освітніх цілях

Та головна сфера, де знайшла своє покликання доповнена реальныcть - це, звичайно, розваги. У 2016 м студія Nyantic випустила спадкоємицю своєї гри Ingress і найголовнішу AR-гру, ймовірно, на багато років вперед - Pokemon Go, інтерфейс якої зображено на рис. 1.11. Доповнена реальність, геотрекінг і популярний всесвіт - все склалося настільки вдало, що Pokemon Go скачали понад сто мільйонів людей. Гра швидко стала феноменом і почала збирати навколо себе скандали. Pokemon Go унікальна ще й тим, що змусила мільйони людей гуляти на свіжому повітрі.

Такі компанії як Lego і Disney активно ведуть розробку ігор з використанням AR, а наміри до них приєднатися висловили практично всі великі виробники іграшок. Дослідницькі групи вже зайнялися збором даних про те, як маленькі діти взаємодіють з іграми і додатками доповненої реальності, і яким чином це впливає на їх сприйняття реального світу. Можливо, в майбутньому найцікавіші ідеї з розвитку технології будуть звучати від тих, для кого ця сама технологія була просто частиною дитинства. Зокрема на презинтаці WWDC (Worldwide Developers Conference) компанія Lego презентувала концепт взаємодії людини з побудованим у доповненій реальності, навколо одного реального будинку з лего, містом, забраженим на рис. 1.12.

Рис. 1.11. Інтерфейс гри «Pockemon Go»

Рис. 1.12. Інтерфейс взаємодії з віртуальним Лего-містом

Саме розваги сьогодні розвивають дослідницьку базу доповненої реальності. А завдяки колосальним обсягами даних, добровільно переданих людьми компаніям-розробникам, технологія в зв'язці з машинним навчанням роблять кроки в бік більш серйозних областей.

І якщо системи наведення в бойових винищувачах, дронах і танках для армії - це сьогодні справа звичайна, тому що саме з ранніх систем доповненої реальності для льотчиків і розвивалися інші військові проекти в цій галузі. Наприклад, просунуті системи доповненої реальності для піхоти, які будуть впроваджуватися вже через пару років, є новинкою.

В американській армії вже сьогодні використовується система HUD 1.0, інтерфейс якої зображено на рис. 1.13. Це сильно вдосконалений прилад нічного бачення, який також виконує функції тепловізора і проектує в монокль на шоломі цілевказувач, що показує куди потрапить куля за поточного положення зброї.

Більш примітивні аналоги таких систем вже більше п'яти років доступні на ринку. Балістичний калькулятор від компанії TrackingPoint, фактично замінює снайперу, ну або будь-якому охочому, напарника-споттера.

На черзі - HUD 3.0, який повинен вийти в наступному році. Він буде мати можливість накладати на реальну картинку повністю цифрові шари місцевості, моделі будівель, плани поверхів, позиції ворогів і навіть самих ворогів. А це вже заявка на здешевлення військових навчань. Військові ігри обходяться державним бюджетам в колосальні суми щороку, а за допомогою систем доповненої реальності солдати зможуть тренуватися з умовним противником не залишаючи меж бази.

Впровадження нових інформаційних технологій не обходить стороною і таку галузь, як будівництво. Не стало винятком і поява технології доповненої реальності в архітектурі, плануванні і проектуванні в будівництві. Застосування технологій доповненої реальності допомагає візуалізувати різні процеси, в тому числі технологічно складні, показати передбачуваний кінцевий результат, як зображено на рис. 1.14, спланувати конструктивні зміни, внести ці зміни на стадії планування і проектування. Також за допомогою доповненої реальності можна користуватися безліччю допоміжних інструментів за допомогою звичайного смартфона. А використовуючи хмарні

технології та 4G зв'язок, можна вивантажувати данні про, наприклад, розмітку на підлозі або зміни у плануванні кімнати - у реальному часі.

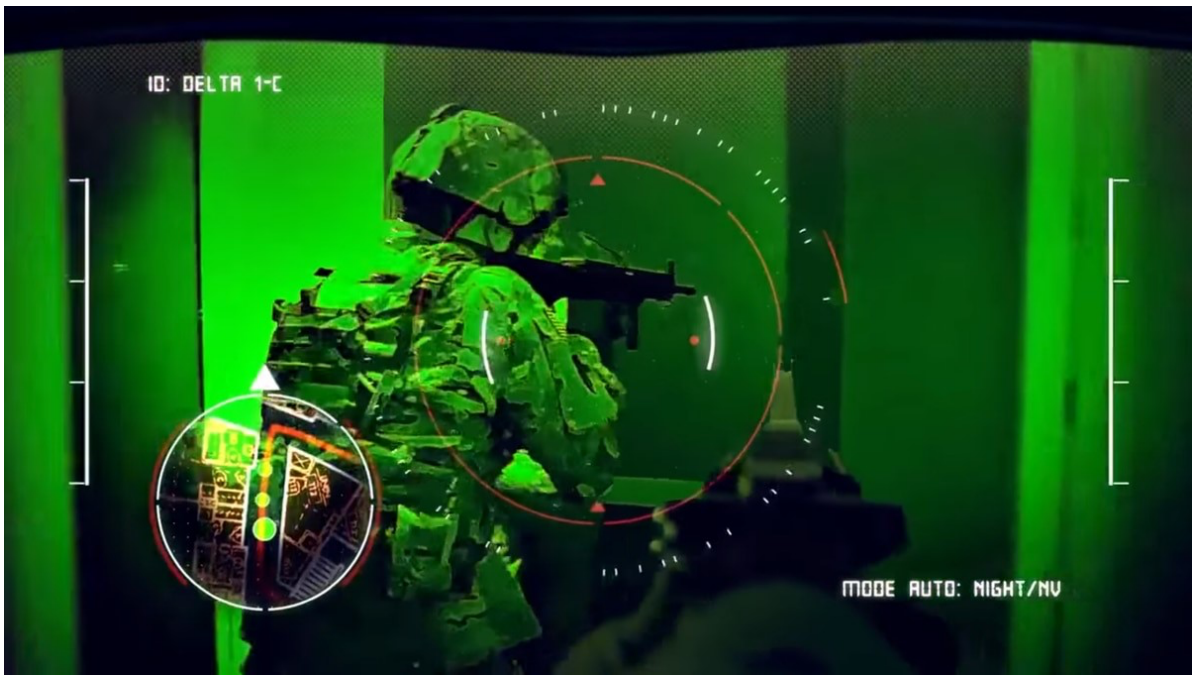

Рис. 1.13. Інтерфейс військової системи для піхоти «HUD 1.0»

Рис. 1.14. Інтерфейс программи для планування будівельних робіт

ВИСНОВКИ ДО РОЗДІЛУ 1

У даному розділі були досліджені ранні прототипи, а також сучасні реалізації приладів та програм для взаємодії з доповненою реальністю. Також були розглянуті сфери, де застосовується технологія доповненої реальності.

Так як усі ці напрямки дуже відрізняються одне від одного, потрібно дослідити, які саме технології та засоби розробки допомагають втілити ці проекти в життя.

РОЗДІЛ 2.

2. ТЕХНІЧНІ ТА ПРОГРАМНІ ЗАСОБИ ДОПОВНЕНОЇ РЕАЛЬНОСТІ

2.1. Технічні засоби взаємодії з доповненою реальністю

Для роботи з додатками доповненої реальності найчастіше використовуються портативні пристрої: смартфони та планшети.

Їх об'єднують чотири складові:

1. Пристрій вводу виводу;

2. Дисплей;

3. Процесор;

4. Пристрій відслідковування.

Розрізняють такі типи дисплеїв, які використовують для відображення доповненої реальності:

1. Ручні дисплеї - смартфони і планшети. Такі пристрої використовують принцип «прозорого екрану» - виводять на екран інформацію, що доповнює реальний об'єкт, яку бачив би користувач, якби екран був повністю прозорим. Є поширеними і загальнодоступними;

2. Head mounted displays (HMD) - пристрій, який кріпиться на голові користувача. До таких пристроїв можна віднести окуляри, контактні лінзи, віртуальний екран на сітківці, технологію EyeTap. Дані пристрої виводять зображення реального світу і додаткових віртуальних об'єктів в полі зору користувача на вбудований екран. Контактні лінзи доповненої реальності зараз знаходяться в розробці. Такі лінзи будуть містити вбудовані дисплеї, електричні схеми, пристрої для бездротової передачі даних. Віртуальний екран на сітківці також знаходиться на стадії розробки в лабораторії Human Interface Technology Вашингтонського університету. Зображення проектується безпосередньо на сітківку ока.

Технологія EyeTap або «Окуляри другого покоління» побудована на такому принципі: пристрій перехоплює світлові промені, що проходять через зіницю, і замінює їх на контрольовані комп'ютером, як зображено на рис. 2.1;

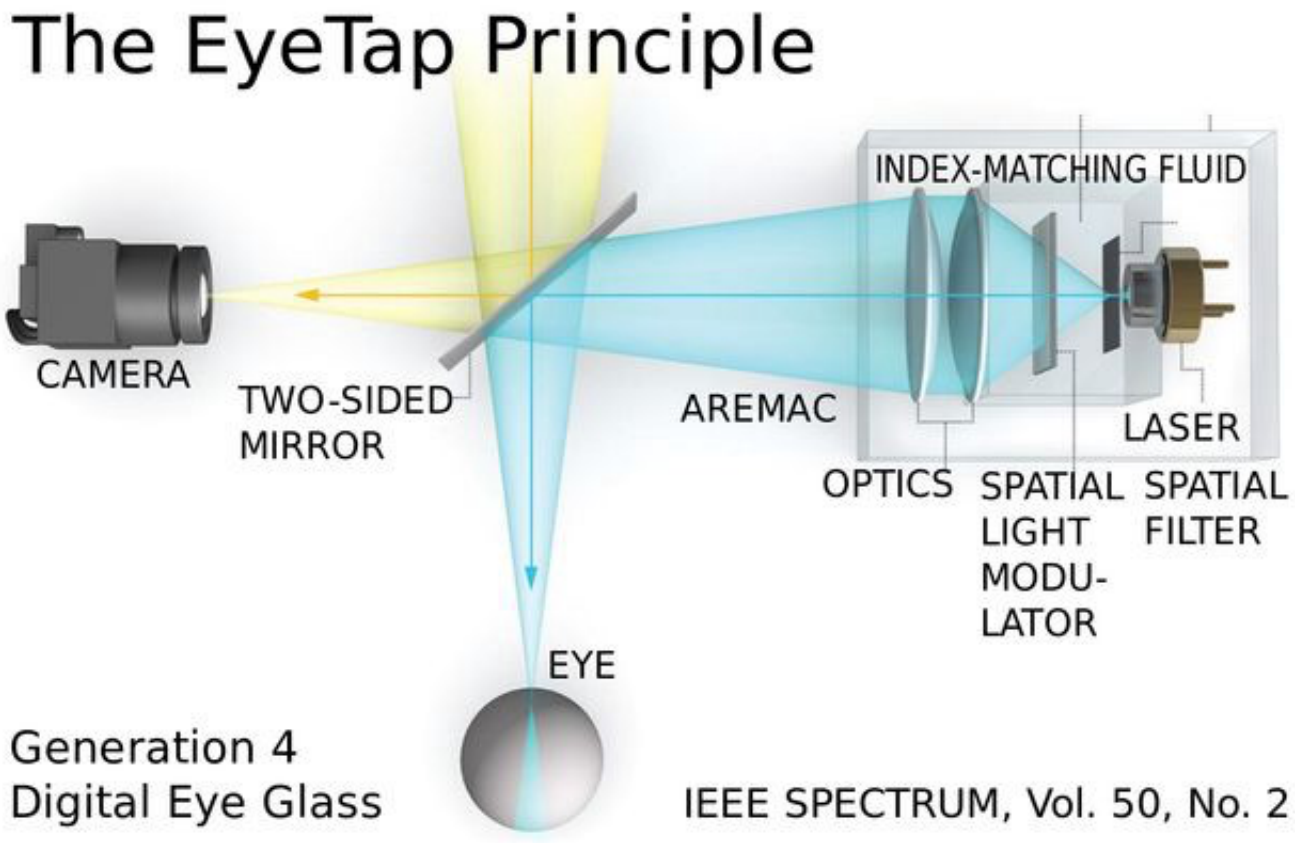

Рис. 2.1. Принцип роботи технології «EyeTap»

3. Просторові дисплеї - голограми, відеопроектори. Головна відмінність таких дисплеїв полягає в тому, що вони не призначені для використання конкретним користувачем. Інформація, що відображається доступна відразу декільком користувачем і представлена у вигляді голограми.

Пристрої введення діляться на системи розпізнавання мови в режимі реального часу, які перетворюють слова в команди і системи розпізнавання жестів, що працюють за допомогою сенсорних пристроїв або оптичних детекторів.

У системах доповненої реальності використовуються наступні пристрої відстеження: цифрові камери, оптичні сенсори, GPS, твердотільні компаси, гіроскопи, бездротові сенсори і т.д. Найважливіше визначити положення і

орієнтацію в просторі голови користувача, а потім положення й орієнтацію пристрою введення.

Системи доповненої реальності повинні мати достатній обсяг оперативної і відео пам'яті, а також володіти потужним процесором для обробки зображень з камери. Сучасні пристрої мають достатню потужність для таких завдань, тому в даний час кожен власник смартфона може дозволити собі використання доповненої реальності.

2.2. Класифікація систем доповненої реальності

По переданій інформації системи доповненої реальності поділяються на:

1. Візуальні, в основі яких лежить зорове сприйняття інформації людиною. Даний вид систем доповненої реальності є найбільш популярним, оскільки візуальне сприйняття є найбільш простим і інформативним;

2. Аудіо - орієнтовані на слухове сприйняття. Використовуються в навігації;

3. Аудіовізуальні. Є об'єднанням двох попередніх систем. Аудіоінформція в даних системах несе допоміжний характер.

Системи доповненої реальності отримують інформацію з навколишнього середовища, тому кожна з них має набір пристроїв приймачів інформації. За типом приймачів системи доповненої реальності діляться на:

1. Оптичні або відеоінформаційні. Обробляють відеосигнал, отриманий з камери, потім на підставі отриманої відеоінформації доповнюють реальний світ віртуальними об'єктами. Оптичні системи діляться на маркерні, безмаркерні та системи просторового трекінгу. Маркерні системи здійснюють пошук маркерних зображень і їх розпізнавання. Приклади маркерного зображення наведено на рис. 2.2.

Рис. 2.2. Приклади маркерів для системи доповненої реальності

Безмаркерні оптичні системи реалізують більш складні алгоритми розпізнавання, використовують не спеціальні маркери, а звичайні зображення або об'ємні тіла.

У системах просторового трекінгу поточний стан доповнюється об'єкта в просторі визначається на основі безперервного аналізу відеопотоку, що надходить з відеокамери в режимі реального часу, для них не потрібні спеціальні зображення;

2. Геопозиційні - орієнтуються на сигнали систем глобального географічного позиціонування ГЛОНАСС або GPS;

3. Комбіновані - використовують як геопозиційні так і оптичні дані, тим самим досягається максимальна точність положення об'єктів доповненої реальності в просторі.

Також системи доповненої реальності можна розрізняти за ступенем взаємодії з користувачем. В одних системах користувач грає вторинну роль, лише спостерігаючи за реакцією системи на зміни в навколишньому середовищі. В інших же потрібна активна участь користувача - управління здійснює користувач самостійно. Таким чином, системи діляться на:

1. Автономні - не вимагають втручання користувача. Завдання таких систем полягає в наданні довідкової інформації про різні об'єкти. Наприклад, автономні системи доповненої реальності застосовуються в медицині під час складних операцій;

2. Інтерактивні - засновані на діалозі з користувачем. У таких системах використовується пристрій введення інформації, наприклад, екран смартфона або планшетного комп'ютера.

Далі системи доповненої реальності можна розділити за ступенем мобільності:

1. Стаціонарні - системи, які мають на увазі роботу у фіксованому місці. Найчастіше є широкоформатними екрани, обладнаними камерами високої роздільної здатності. При переміщенні такі системи частково або повністю

втрачають свою працездатність, але при цьому в статичному стані демонструють більш реалістичну візуалізацію;

2. Мобільні системи можуть без проблем переміщатися в просторі. До них відносять смартфони і планшетні комп'ютери.

Таким чином, приналежність системи до того чи іншого типу визначається її функціоналом. Так, наприклад, студентам-медикам підійдуть стаціонарні системи, так як симуляція складних операцій не має на увазі пересування віртуального пацієнта в просторі. А для будівельників більш зручними будуть мобільні системи, так як їм може стати в нагоді навігаційна система, різноманітні вимірювальні прилади та візуалізація плану будівлі.

2.3. Алгоритм розпізнавання маркерів

Маркер - це об'єкт, розташований в навколишньому просторі, який відстежується і аналізується спеціальним програмним забезпеченням для подальшого зображення на ньому віртуальних об'єктів. Відповідно до положення маркера в просторі, програма може достатньо точно помістити на нього віртуальний об'єкт, в результаті чого буде досягнуто ефект його фізичного присутності в просторі. За допомогою додаткових графічних фільтрів і високоякісних моделей, віртуальний об'єкт може стати практично реальним і важко відмінним від інших елементів інтер'єру або екстер'єру.

Найчастіше в ролі маркера виступає аркуш паперу з деяким спеціальним зображенням. Тип малюнка може варіюватися досить сильно і залежить від алгоритмів розпізнавання зображень. Взагалі, різноманіття маркерів досить велике: ними можуть бути і геометричні фігури простої форми (наприклад, коло, квадрат), і об'єкти у формі прямокутного паралелепіпеда, і навіть очі та обличчя людей.

Обкладинку журналу додаток розпізнає по простій формі з прямими кутами і конкретним малюнку, і буде відслідковувати її положення в просторі, відзначаючи зміщення відносно фону. У цьому випадку сама обкладинка і є маркер, як зображено на рис. 2.3.

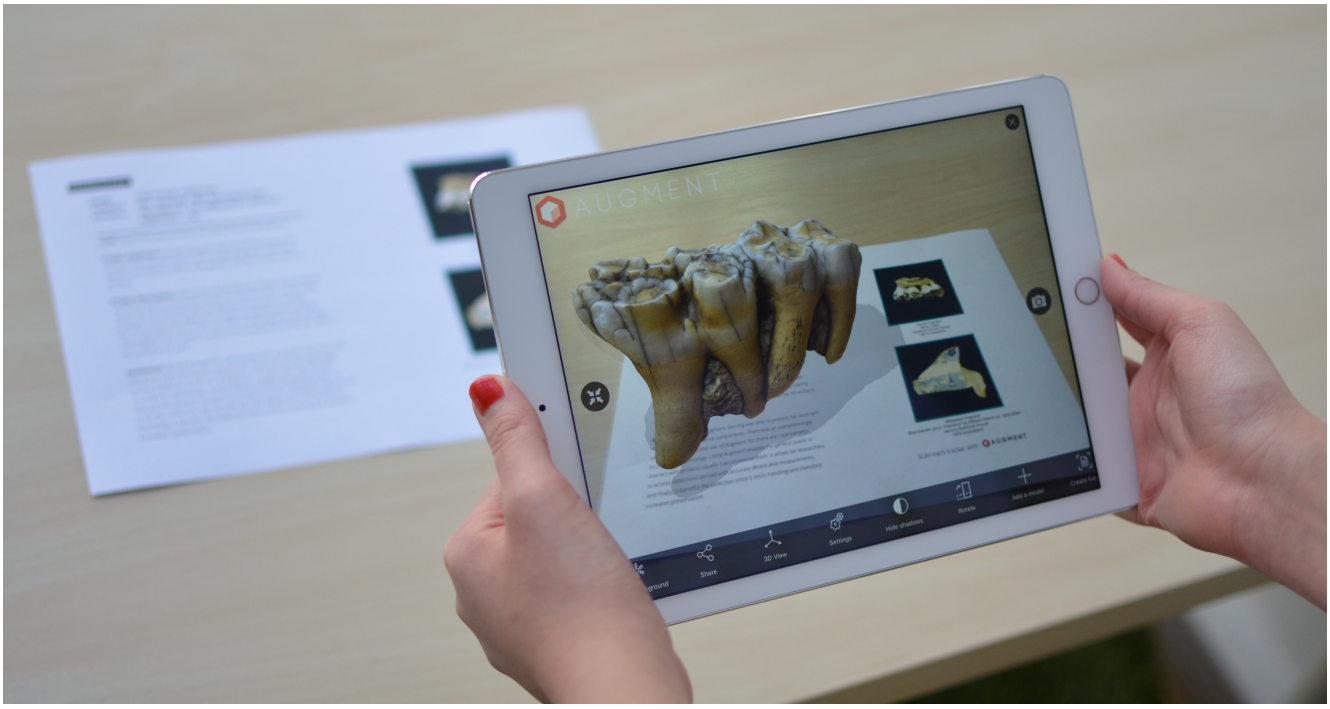

Рис. 2.3. Роздрукована стаття виступає маркером для програми з доповненою реальністю

Зі спеціальними маркерами все йде ще простіше. Припустимо, ми хочемо приміряти автомобілю нові диски. Для цього нам досить наклеїти на диски QR-мітки і система автоматично зрозуміє, що саме в цих місцях слід вставляти в картинку зображення нових коліс. Ще один приклад: ми кладемо мітку, приклад яких наведено на рис. 2.4, на підлогу і додаток розуміє, що ця площина і є підлога, і розмістить на ньому довільні об’єкти.

Рис. 2.4. Різні спеціальні маркери виступають мітками для розміщення віртуальних об’єктів

2.3.1. Комп’ютерний зір

Теорія комп'ютерного зору є основоположною для розвитку технологій доповненої реальності, і перш за все в галузі використання маркерів. Основний напрямок даної дисципліни - це аналіз і обробка зображень, в тому числі і відеопотоку. Алгоритми комп'ютерного зору дозволяють виділяти ключові особливості на зображенні (кути, межі області), проводити пошук фігур і об'єктів в реальному часі, виконувати 3D реконструкцію з кількох фотографій і багато іншого.

В області доповненої реальності алгоритми комп'ютерного зору використовуються для пошуку в відеопотоці спеціальних маркерів. Залежно від завдання в якості маркера можуть виступати як спеціально сформовані зображення, так і обличчя людей. Після знаходження маркера в відеопотоці і обчислення його розташування, з'являється можливість побудови матриці проекції і позиціонування віртуальних моделей. За допомогою них можна накласти віртуальний об'єкт на відеопотік таким чином, що буде досягнутий ефект присутності. Основна складність якраз і полягає в тому, щоб знайти маркер, визначити його місцерозташування в кадрі і спроектувати відповідним чином віртуальну модель.

За останні роки було розроблено багато методів обробки зображень та знаходження на них різних об’єктів. Розглянемо ті, що застосовують під час створення доповненої реальності.

2.3.1.1. Генетичні алгоритми

Генетичні алгоритми - це евристичні алгоритми пошуку, які використовуються для вирішення завдань оптимізації та моделювання шляхом випадкового підбору, комбінування і варіації шуканих параметрів із використанням механізмів, що нагадують біологічну еволюцію.

У комп'ютерному зорі вони використовуються для пошуку об'єкта деякого заданого класу на статичному зображенні або відео потоці. Спочатку

необхідно провести навчання алгоритму за допомогою двох різних наборів зображень:

1. "Хороші" - містять потрібний об'єкт.

2. "Погані" - помилкові зображення без шуканого об'єкта.

При цьому для навчання використовується велика кількість зображень, і чим їх більше - тим краще буде працювати сам алгоритм. Для кожної картинки проводиться виділення різних ключових особливостей: кордони, лінії, центральні елементи. По ним проводиться побудова статистичної моделі, яка потім і використовується для пошуку об'єкта на зображенні.

Прикладом використання даного підходу може служити алгоритм розпізнавання облич і очей на відеопотоці, у якому використовуються примітиви Хаара, зображені на рис. 2.5. За допомогою них можна виділити області на зображенні з більшою та меншою інтенсивніст'ю пікселів та дати алгоритму зрозуміти що на ньому є, або немає обличчя. Поступово навчаючи алгоритм, можна домогтися високих результатів знаходження заданого класу об'єктів. Однак необхідність навчання якраз і робить використання генетичних алгоритмів досить проблематичним. Для їх хорошої роботи потрібно значне число різних зображень, як "хороших", так і "поганих", і побудова класифікатора для кожного об'єкта може займати тривалий час.

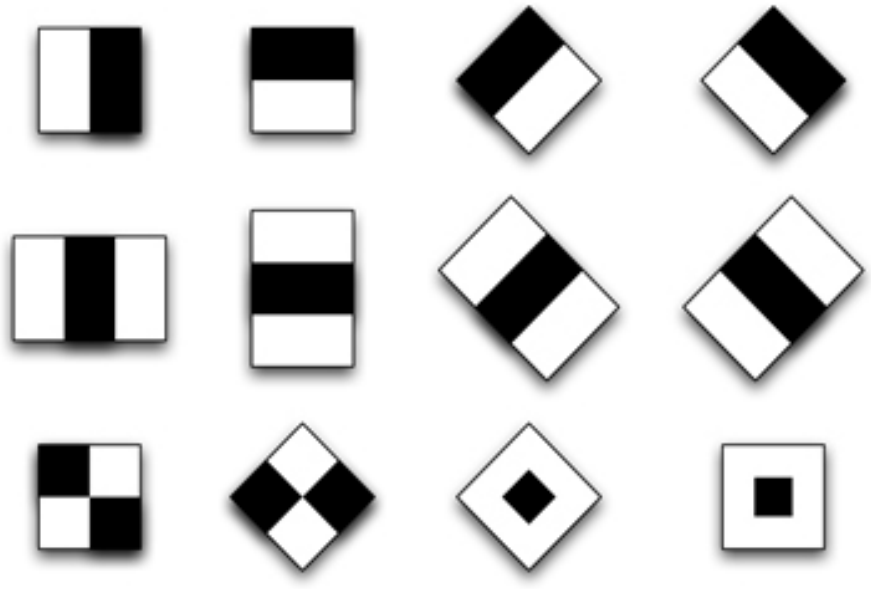

Рис. 2.5. Примітиви Хаара

2.3.1.2. Feature detection

Концепція feature detection (укр. виявлення особливостей) в комп'ютерному зорі відноситься до методів, які націлені на обчислення абстракцій зображення і виділення на ньому ключових особливостей. Дані особливості можуть бути як у вигляді ізольованих точок, так і кривих або пов'язаних областей. Не існує жорсткого визначення того, що таке ключова особливість зображення. Кожен алгоритм розуміє під цим своє (кути, межі, області тощо).

Найчастіше для пошуку маркерів використовуються алгоритми, які виконують пошук і порівняння зображень по ключових точках.

Ключова точка - це деяка ділянка картинки, яка є характерною для заданого зображення. Що саме приймається за дану точку - безпосередньо залежить від алгоритму, який використовується. Приклад ключових точок наведено на рис. 2.6.

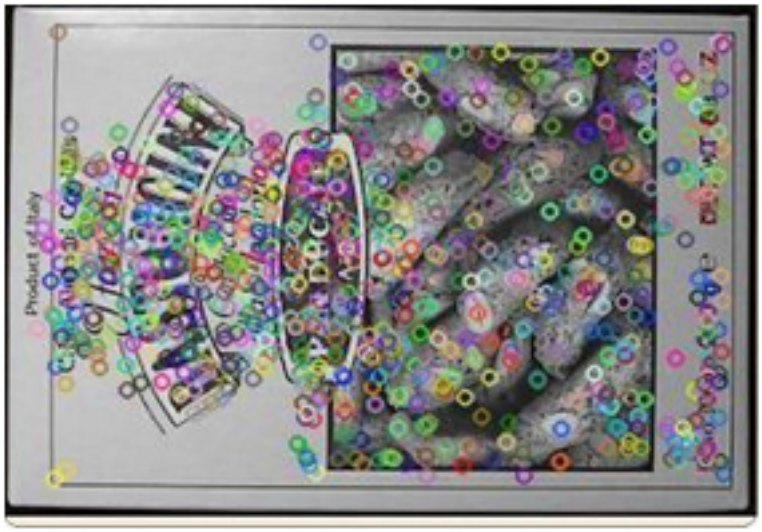

Рис. 2.6. Ключові точки на зображенні

Для їх знаходження і подальшого порівняння використовуються три складові:

1. Детектор - здійснює пошук ключових точок на зображенні;
2. Дескриптор - робить опис знайдених ключових точок, оцінюючи їх позиції через опис навколишніх областей;

3. Матчер - будує відповідності між двома наборами точок.

Спочатку за допомогою детектора проводиться пошук ключових точок шаблонного зображення. Отримані точки потім описуються за допомогою дескриптора. Дана інформація зберігається в окремий файл,або базу данних, щоб не виконувати цей процес повторно. При обробці відеопотоку з метою пошуку заданого шаблону описаний процес виконується для кожного кадру,за винятком збереження даних. Для встановлення відповідності між ключовими точками і дескрипторами застосовується матчер, як показано на рис. 2.7.

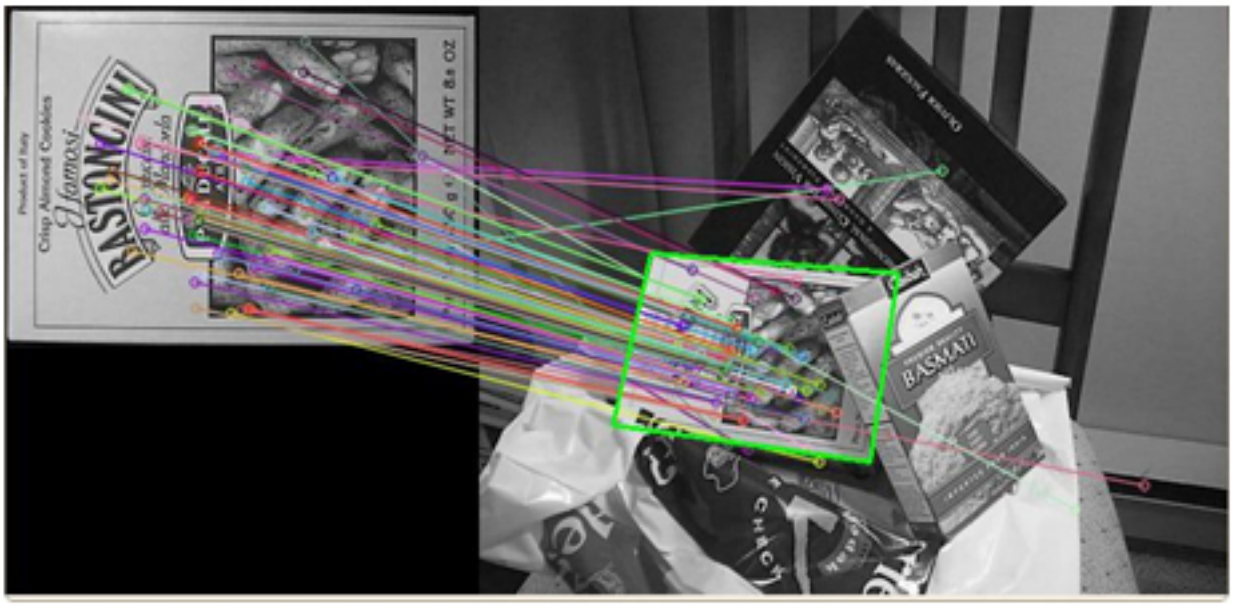

Рис. 2.7. Встановлення відповідності між точками шаблона та зображенням

Природно припустити, що різні алгоритми працюють з різною швидкістю і ефективністю. В умовах застосування їх для побудови доповненої реальності необхідно використовувати тільки ті, які показують високу швидкість роботи при досить хорошій якості відстеження позицій ключових точок. В іншому випадку ми можемо отримати помітні відставання у відеопотоці.

Для підвищення швидкодії алгоритмів feature points detection застосовуються різні способи фільтрації точок, щоб мінімізувати їх число і прибрати зовсім погані поєднання. Таким чином, можна домогтися не тільки підвищення швидкості роботи алгоритмів, але і якості трекінгу маркерів. Але

маркери скрізь не наліпити, а зробити унікальний маркер під кожну ситуацію і уніфікувати всю систему занадто складно.

2.4. Алгоритм SLAM

Тут на виручку приходить SLAM - метод одночасної локалізації та побудови карти, який використовується для побудови карти в невідомому просторі з одночасним контролем поточного місцезнаходження і пройденого шляху. Одночасна локалізація і картографія (SLAM) являє собою ряд складних обчислень і алгоритмів, які використовують дані датчиків, щоб побудувати карту невідомого середовища при його використанні в той же час, щоб визначити, де він розташований. Алгоритми SLAM поєднують дані з датчиків для визначення положення кожного датчика або обробляють дані, отримані від них, і будують карту навколишнього середовища. SLAM стає все більш важливою темою у спільноті комп'ютерного зору і отримує особливий інтерес у галузях, включаючи доповнену та віртуальну реальність.

Перш за все, існує величезна кількість різних апаратних засобів, які можна використовувати. По-друге, SLAM більше схожий на концепцію, ніж на один алгоритм. Існує багато етапів, що стосуються SLAM, і ці різні кроки можуть бути реалізовані з використанням декількох різних алгоритмів. Сфери застосування цього алгоритму досить широкі. Найближчими до теми є застосування у:

1. VR, користувачі хотіли б взаємодіяти з об'єктами у віртуальному середовищі без використання зовнішніх контролерів;

2. AR рендеринг об'єкта має відповідати реальному 3D-середовищу, особливо коли користувач рухається;

3. Навіть більш важливим є те, що в автономних транспортних засобах, таких як безпілотні літальні апарати, транспортний засіб повинен знайти своє місце розташування в 3D-середовищі.

Існує багато методів SLAM відповідно до реалізації та використання: EKF SLAM, FastSLAM, SLAM на основі графіків, топологічний SLAM та багато

іншого. Але тут я збираюся розглянути тільки 2 методи із яких Visual SLAM цікавіше з точки зору AR / VR / MR.

Visual SLAM: монокулярні, стереофонічні або камери з дальнім зумом - первинний тип датчика.

Злиття сенсорів: об'єднання інформації від декількох датчиків в єдине рішення

2.4.1. Visual SLAM

Visual SLAM зараз дуже добре підходить для відстеження в невідомих середовищах, приміщеннях, просторах і 3D-моделях або об'єктах реального світу, де основний режим сканування здійснюється за допомогою камери - оскільки він представляє найбільший інтерес у контексті доповненої реальності.

Потреба у знаходженні положення камери та побудові карти, коли жодна з них не відома, відрізняє проблему SLAM від інших завдань. Наприклад, відстеження на основі маркера не є SLAM, оскільки зображення маркера (аналогічне карті) відомо заздалегідь. 3D-реконструкція з фіксованою фотокамерою не є SLAM тому, що в той час як карта оновлюється, позиції камер вже відомі. Завдання в SLAM полягає в тому, щоб знайти як позицію камери, так і побудувати структуру карти, хоча спочатку не знаючи нічого.

2.4.2. Злиття сенсорів

По своїй суті цей метод нічим не відрізняється від Visual SLAM, проте на допомогу камері приходять інші сенсори, які встановлені на ґаджеті. На приклад це можуть бути:

1. Датчики RGB - допомагають камері бачити так звані глибинні пікселі;
2. IMU - інерційний вимірювальний пристрій, який мірює швидкість приладу, його орієнтацію у просторі та гравітаційні сили, що діють на нього;
3. GPS - допомагає приладу орієнтуватися у просторі за допомогою супутників.

2.4.3. Принцип роботи алгоритму SLAM

SLAM - це проблема з тим, що було раніше: курка чи яйце.

1. Карта необхідна для оцінки місцезнаходження;

2. А оцінка місцезнаходження необхідне для побудови мапи.

SLAM схожий на людину, яка намагається знайти вихід з невідомого місця. По-перше, людина дивиться навколо, щоб знайти знайомі маркери або знаки. Після того, як людина знаходить знайомий орієнтир, він / вона може зрозуміти де знаходиться. Якщо особа не розпізнає орієнтири, її можна буде позначити як заблукавшу. Однак, чим більше людина спостерігає за навколишнім середовищем, тим більше орієнтирів людина розпізнає і починає будувати розумовий образ, або карту цього місця.

Схема, зображена на рис. 2.8, показує як працює алгоритм SLAM.

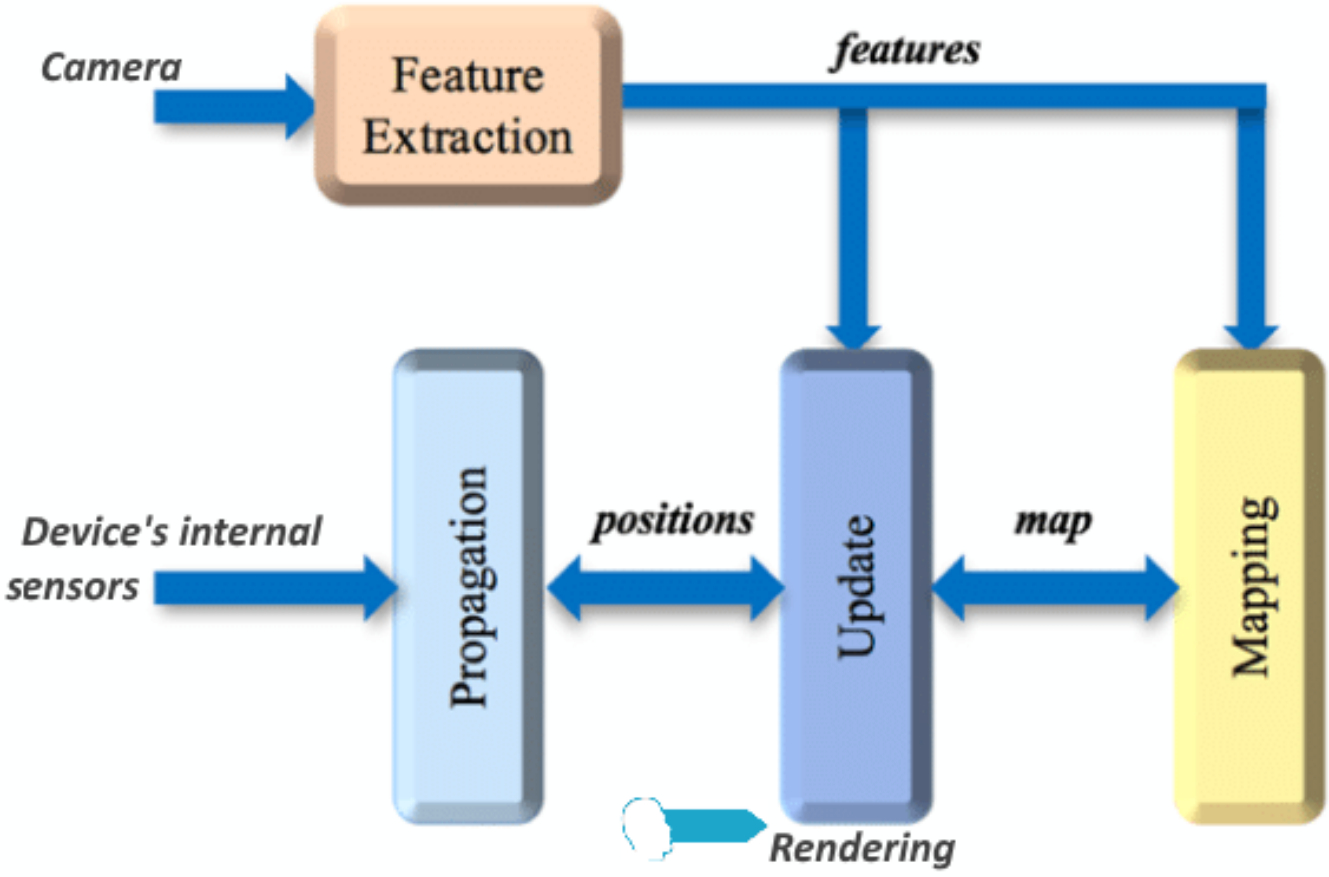

Рис. 2.8. Схема роботи алгоритму SLAM

1. Внутрішні датчики або IMU складається з гіроскопа та інших сучасних датчиків для вимірювання кутової швидкості і акселерометрів для вимірювання прискорення в трьох осях і переміщення користувача;

2. Головним завданням блоку розмноження є інтеграція точок даних IMU і створення нової позиції. Однак, оскільки апаратні засоби IMU зазвичай мають упередженість та неточності, ми не можемо повністю покладатися на данні блоку розмноження;

3. Щоб виправити проблему смикання, ми використовуємо камеру для зйомки кадрів з фіксованою швидкістю, як правило, при 60 FPS. Сучасні пристрої мають спеціальну камеру з телеоб'єктивом;

4. Кадри, зняті камерою, можуть подаватися на блок знаходження ключових точок, який витягує корисні кутові функції і генерує дескриптор для кожної функції;

5. Отримані ключові точки можуть потім бути подані в блок побудови мапи, щоб розширити карту, яку досліджує пристрій;

6. Крім того, виявлені ключові точки надсилатимуться до блоку оновлення, який порівнює їх з картою. Якщо виявлені точки вже існують на карті, пристрій оновлень може вивести поточну позицію пристрою з відомих точок на карті;

7. Використовуючи цю нову позицію, модуль оновлення може виправити смикання, що вводиться блоком розмноження. Також блок оновлення оновлює карту новими ключовими точками.

Під час переміщення та сдвигу камери відбувається наступне:

1. Порівняння даних локального зсуву і особливостей.

Тепер у нас з одного боку є результати локального зсуву, на основі якого ми можемо обчислити нове положення пристрою. З іншого боку - ми так само можемо визначити позицію пристрою щодо особливостей сцени. В силу похибок сенсорів і використовуваних алгоритмів дані позиції не збігатимуться між собою і, швидше за все, також і з реальним місцем розташування

пристрою. Отримана різниця повинна бути відображена у всіх попередніх вимірах особливостей як коригувальний коефіцієнт.

2. Оновлення загального уявлення про карту світу і траєкторії пристрою.

На основі знову отриманих даних проводиться уточнення поточних уявлень про світ - перерахунок позицій пристрою , особливостей і їх ймовірностей. Додатково проводиться пошук замикань - ситуацій, коли пристрій повертається в ті місця, де вже побував. Після всіх обчислень оновлюється загальне уявлення світу - положення пристрою і координати особливостей.

На основі глобальної структури, що зберігає всі переміщення пристрою і уявлення світу в будь-який момент часу можливо відтворити карту світу з траєкторією руху пристрою. Для візуалізації використовуються сторонні засоби, які до методів SLAM особливого відношення не мають, як зображено на рис. 2.9.

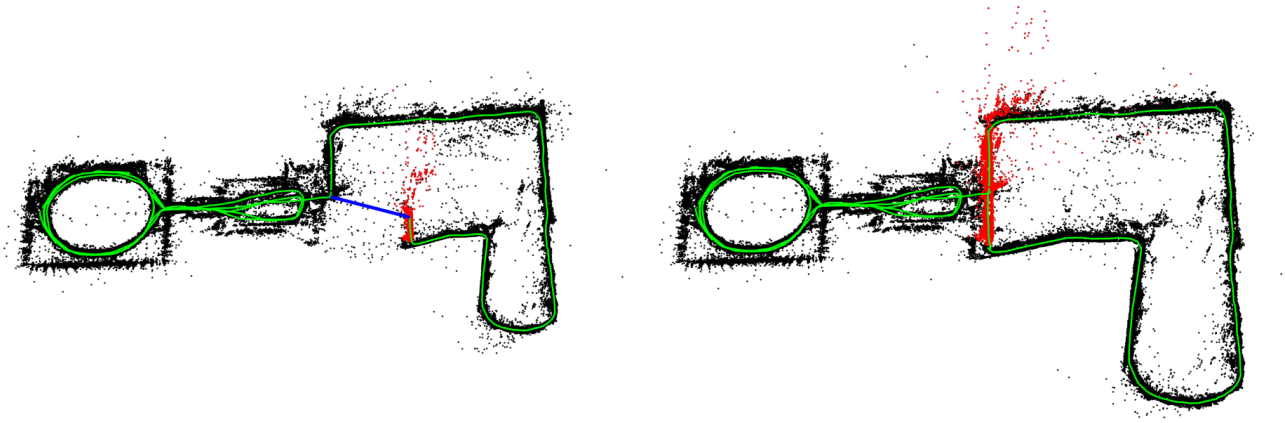

Рис. 2.9. Приклад побудованої мапи навколишнього середовища за допомогою SLAM

Також важливий момент - середовище вважається статичним. Це означає, що якщо пристрій пройшов повз стіл, ніхто за його спиною цей же самий стіл рухати не стане. В кадрі - в розглядаємому сенсорами просторі - жодного стороннього руху немає - ніхто в кадрі не ходить. Освітлення радикально не змінюється - тобто тіні по стінах не стрибають. Такі умови складно

гарантувати в природному середовищі - шелест листя або кішка запросто може дезорієнтувати пристрій в просторі. Звичайно не так запросто - і відсікати динамічну частину можна, і робити поправку на неї. Однак, якщо пофантазувати на тему марсоходів - то навіть у теперішньому вигляді методи SLAM можуть знайти своє застосування. Зокрема, робот-пилосос - яскравий приклад застосування даної методики в 2D-просторі. Коли пилосос прибирає у кімнаті, він запам'ятовує положення предметів у ній і якщо забрати той самий стіл з кімнати, пилосос просто оновить мапу і збереже нову. Існують експериментальні реалізації даного завдання і в 3D, як от для квадрокоптерів, як показано на рис. 2.10.

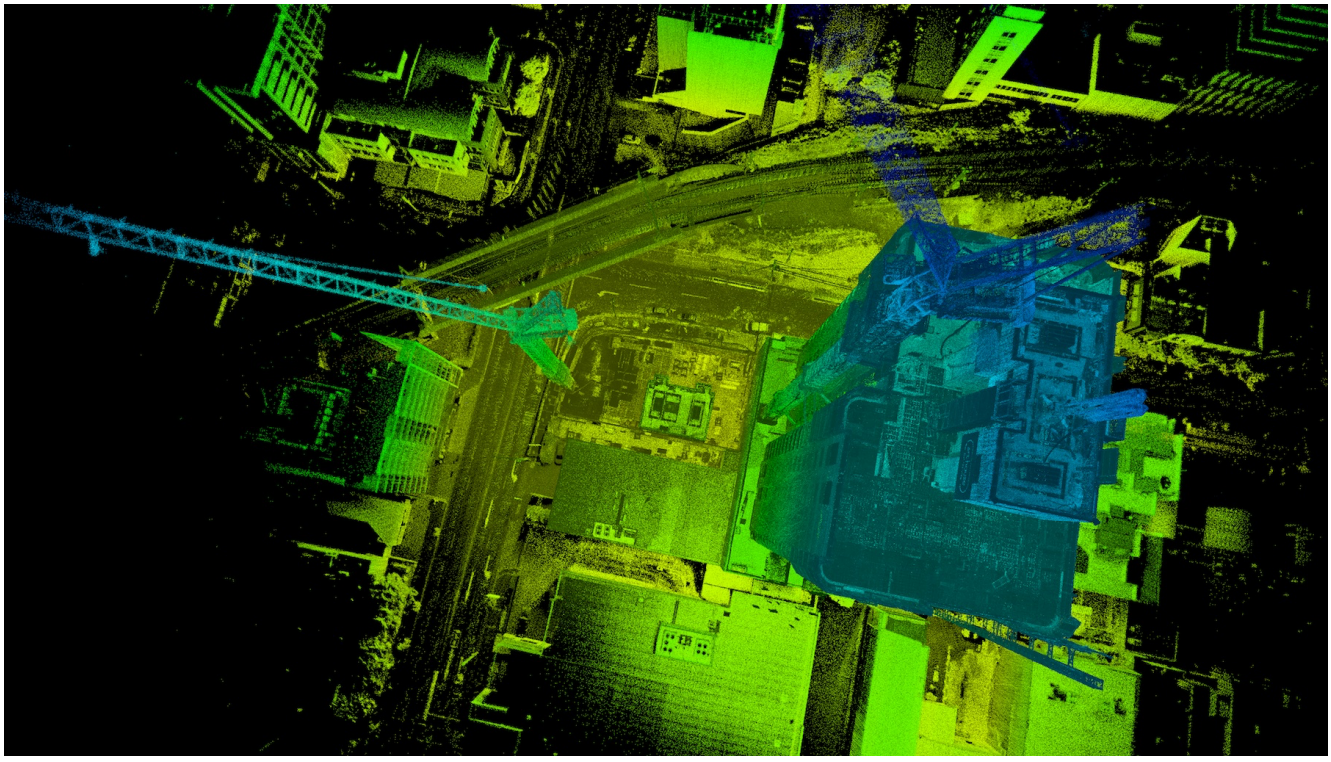

Рис. 2.10 Комбіноване зображення того, що бачить автономний квадрокоптер і реальної картини з камери

2.5. Google ARCore

2.5.1. Історія появи ARCore

Історія доповненої реальності на Android починається в 2014 році з проекту Tango. Він не завоював широку популярність на ринку, оскільки його робота сильно залежала від сенсора глибини - пристрою в смартфоні, який розраховує відстань на підставі відбитої довжини інфрачервоної хвилі. Сенсор дозволяв розміщувати девайс у віртуальний простір, що є відображенням реального. Однак у нього були серйозні обмеження. Так максимально доступна відстань становила всього 4 метри. А відстань, при відбитті від яскравих або дзеркальних поверхонь інфрачервоної хвилі, взагалі не будувалася.

В результаті компанія Google поставила за мету створити доповнену реальність на Android, виключивши залежність від апаратних засобів. Так з'явився ARCore. Для його роботи не потрібен сенсор глибини. Таким чином, досвід, отриманий в процесі роботи над Tango, для нього не знадобився. Але послужив хорошим стартовим майданчиком. На сьогодні ARCore має близько 100 мільйонів потенційних користувачів.

2.5.2. Відстеження руху

Коли ваш мобільний телефон переміщається по навколишньому простору, ARCore об'єднує візуальні дані з камери пристрою, щоб оцінити становище і орієнтацію об'єктива щодо часу і простору. У цій категорії слід виділити різні калібрування. Перелічимо основні:

1. Оптична:

1.1 Pinholde model - математична модель описує відношення між координатами точки в тривимірному просторі з її проекцією на полотно, а також Field of View (FoV) - модель описує викривлення перспективи зображення;

1.2 Фотометричне калібрування - карта інтенсивності кольорів;

2. Моделювання на основі інерції:

2.1 Вимірюється прискорення - не дистанція і не швидкість;

2.2 Моделювання на основі інерції не зводиться до категорії "так чи ні". Це - більше статистика, необхідна для конкретного випадку використання.

Втім, існує температурна проблема при заводському калібруванні IMU. Різні виробники виготовляють його за різних температур, тому дані на різних девайсах можуть відрізнятися. IMU зображено на рис. 2.11.

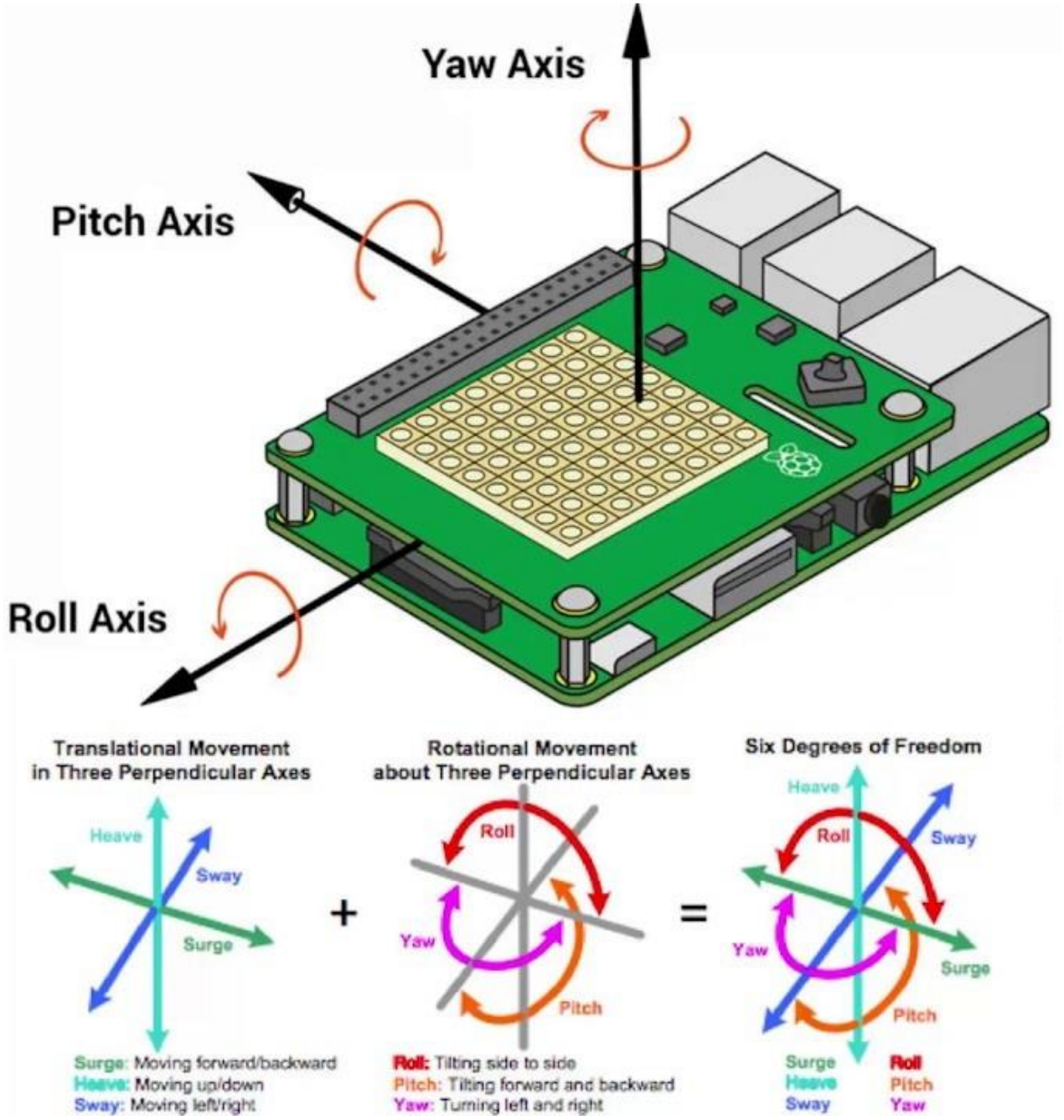

Рис. 2.11. Схематичне зображення IMU

2.5.3. Аналіз навколишнього середовища

ARCore шукає кластери характерних точок, які, як виявляється, лежать на одних і тих самих горизонтальних поверхнях і роблять їх доступними для вашого застосування в вигляді площин.

В основі розуміння оточення лежить технологія SLAM. SLAM карта є графіком 3D точок, які представляють собою розріджену хмару, де кожна позначка відповідає координатам оптичного об'єкта (наприклад, кут таблиці).

Також, як і з вимірами на основі прискорення, SLAM спирається на карту точок, які можуть бути більш-менш надійними. Основне завдання SLAM - побудова і оновлення карти невідомого середовища при одночасному відстеженні місцерозташування пристрою всередині неї. Також як і Tango, має проблеми з дзеркальними поверхнями.

2.5.4. Оцінка світла

ARCore може виявити середню інтенсивність зображення камери, так що ви отримуєте можливість висвітлити віртуальні об'єкти так само, як освітлені об'єкти навколишнього середовища, як показано на рис. 2.12.

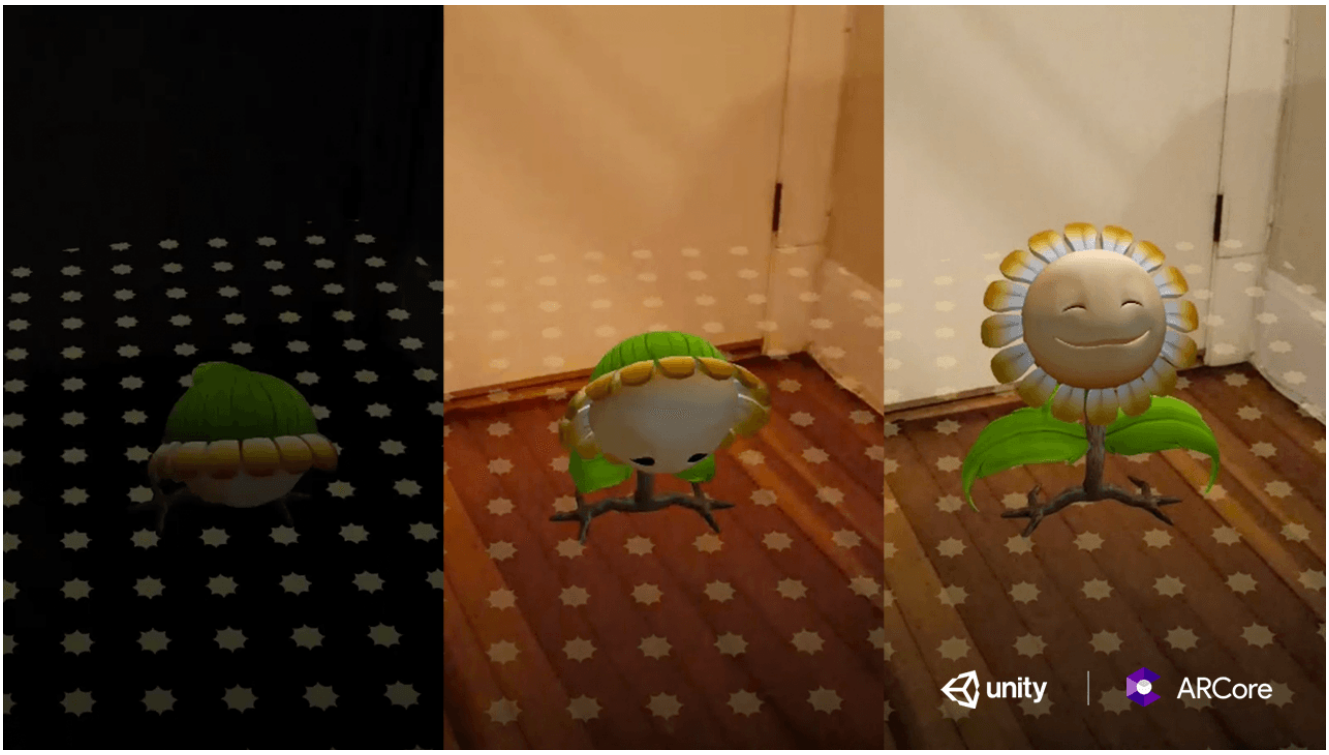

Рис. 2.12. Оцінка світла ARCore

2.5.5. Взаємодія з користувачем

ARCore використовує хіт-тести, щоб прийняти координати (x, y), що відповідають екрану телефону, які забезпечуються натисканням або будь-якою іншою взаємодією, яку потрібно підтримувати додатком, і проєцює промінь у середовище, яке бачить камера, повертаючи будь-які площини або ключові точки, що перетинає промінь, разом з позицією цього перетину в навколишньому просторі. Це дозволяє користувачам вибирати або іншим чином взаємодіяти з об'єктами в середовищі.

2.5.6. Точки орієнтування

Точки орієнтування дозволяють розміщувати віртуальні об'єкти на похилих поверхнях. Коли ви виконуєте хіт-тест, який повертає ключову точку, ARCore буде дивитися на найближчі ключові точки і використовувати їх, щоб спробувати оцінити кут поверхні на даній точці об'єкта. Потім ARCore поверне позицію, яка враховує цей кут. Оскільки ARCore використовує кластери ключових точок для виявлення кута поверхні, поверхні без текстури, такі як білі стіни, можуть бути невірно виявлені.

2.5.7. Прив'язки та відстеження

Позиції можуть змінюватися, оскільки ARCore покращує своє розуміння своєї позиції та навколишнього середовища. Якщо ви хочете розмістити віртуальний об'єкт, вам необхідно визначити прив'язку, щоб переконатися, що ARCore відстежує позицію об'єкта з плином часу. Часто ви створюєте прив'язку на основі позиції, що повертається хіт-тестом, як описано у взаємодії з користувачем.

Той факт, що позиція може змінитися, означає, що ARCore може оновити положення об'єктів навколишнього середовища, таких як площини та ключові точки з плином часу. Площини і точки - це особливі типи об'єкта, які називаються відстежуваними. Як випливає з назви, це об'єкти, які ARCore буде відстежувати з плином часу. Можна закріпити віртуальні об'єкти на конкретні трекери, щоб переконатися, що зв'язок між віртуальним об'єктом і відстежуваним залишається стабільним навіть при переміщенні пристрою. З

цього випливає, що якщо ви поставите віртуальну фігурку на столі, якщо ARCore пізніше налаштує позицію площини стола, фігурка, як і раніше, буде залишатися на поверхні столу.

2.5.8. Доповнені зображення

Доповнені зображення - це функція, яка дозволяє створювати додатки AR, які можуть реагувати на конкретні 2D-зображення, як наприклад фотографія чи обгортка журналу. Користувачі можуть взаємодіяти з AR, коли вони спрямовують камеру свого телефону на певні зображення. Як приклад користувач може навести камеру на обгортку журналу та побачити 3D модель машини. ARCore також відстежує рухомі зображення, такі як, наприклад, реклама на боці рухомого автобусу.

Зображення можна компілювати в автономному режимі для створення колекції зображень, або можна додавати окремі зображення в реальному часі з пристрою. Після цього ARCore виявить ці зображення та їх межі і поверне відповідну позицію, як показано на рис. 2.13.

Рис. 2.13. Приклад доповненого зображення

2.5.9. Спільний доступ

ARCore Cloud Anchor API дозволяє створювати програми для спільних або багатокористувацьких програм для пристроїв Android і iOS.

За допомогою Cloud Anchors один пристрій відправляє прив'язку і прилеглі ключові точки до хмарного сховища для хостингу. Ці прив'язки можна спільно використовувати з іншими користувачами на пристроях Android або iOS в одному середовищі. Це дозволяє програмам відображати ті ж 3D-об'єкти, які прикріплені до цих прив'язок, дозволяючи користувачам мати однаковий досвід одночасно.

2.6. Apple ARKit

Анонсування Apple ARKit на останній конференції WWDC мало величезний вплив на екосистему доповненої реальності. Розробники вперше зустрілися з надійним і широко доступним SDK для доповненої реальності. Немає необхідності використовувати маркери, ініціалізувати камеру глибини або користуватися пропрієтарними інструментами розробки.

2.6.1. Відстеження руху

Відстеження - це основна функція ARKit. ARKit використовує візуальну інерційну одометрію (VIO) для точного відстеження навколишнього світу. Візуальна одометрія означає оцінку 3D-позиції рухомої камери щодо її вихідної позиції, використовуючи візуальні особливості. VIO фіксує дані датчика камери з даними CoreMotion. Ці два входи дозволяють пристрою відчувати, як він рухається в кімнаті з високим ступенем точності без додаткового калібрування. Більш важливим є відсутність необхідного зовнішнього налаштування, відсутність попередніх знань про навколишнє середовище, а також ніяких додаткових датчиків.

2.6.2. Розуміння сцени

Розуміння сцени полягає в можливості визначити атрибути або властивості навколишнього середовища навколо пристрою. За допомогою ARKit, iOS пристрій може проаналізувати сцену, представлену зображенням з камери, і знайти горизонтальні площини в кімнаті. ARKit також використовує

датчик камери, щоб оцінити загальну кількість світла, доступного в сцені, і застосовує правильну кількість освітлення до віртуальних об'єктів. Функція хіт-тестів забезпечує перетин з топологією реального світу, щоб віртуальні об'єкти могли бути розміщені у фізичному світі. Також у версії ARKit 2.0 додали можливість зберігати проскановане середовище у об'єкт ARWolrdMap, який представляє собою відображення фізичного 3D-простору, який може бути виявлений телефоном і використаний для створення ARScene.

2.6.3. Візуалізація

ARKit надає постійний потік зображень з камер, які доповнюють інформацію та розуміння сцени, які можуть бути занесені в будь-який візуалізатор, включаючи інструменти SceneKit, Metal, SpriteKit та сторонні інструменти, такі як Unity та Unreal Engine. У версії ARKit 2.0 візуалізація була покращена і тепер віртуальні об’єкти можут реалістичніше взаємодіяти зі світлом, додано можливість бачити відображення реального навколишнього середовища на віртуальних об’єктах і завдяки кращому відстеженню рухів і розумінню сцени, ніж у конкурентів, можна наближатися впритул до них і не боятися що об’єкт зникне чи почне переміщуватися по сцені.

2.6.4. Виявлення та відстеження зображення

У iOS 11.3 Apple представила функцію виявлення зображень як частину відстеження за навколишнім середовищем. Функція виявлення зображення шукає відомі 2D-зображення у сцені, проте зображення повинні були бути статичними.

Відстеження зображень є розширенням функції виявлення зображень з величезною перевагою - зображення більше не повинні бути статичними. ARKit оцінить позицію і орієнтацію для кожного кадру при 60 кадрах в секунду. Підтримується також багатократне виявлення зображень, що означає, що ARKit може відстежувати, розпізнавати та розміщувати декілька зображень одночасно.

2.6.5. Відстеження об'єктів

Відстеження об'єктів може бути використано для виявлення 3D-об'єктів у ARScene. Так само, як виявлення зображення, ці об'єкти повинні бути статичними. Для цього об'єкти потрібно спочатку сканувати за допомогою програми, як показано на рис. 2.14, а потім зберегти локально. Об'єкти, які скануються, повинні бути добре текстурованими, жорсткими та нерефлекторними. Скануючи об'єкт, ви отримуєте його положення та орієнтацію.

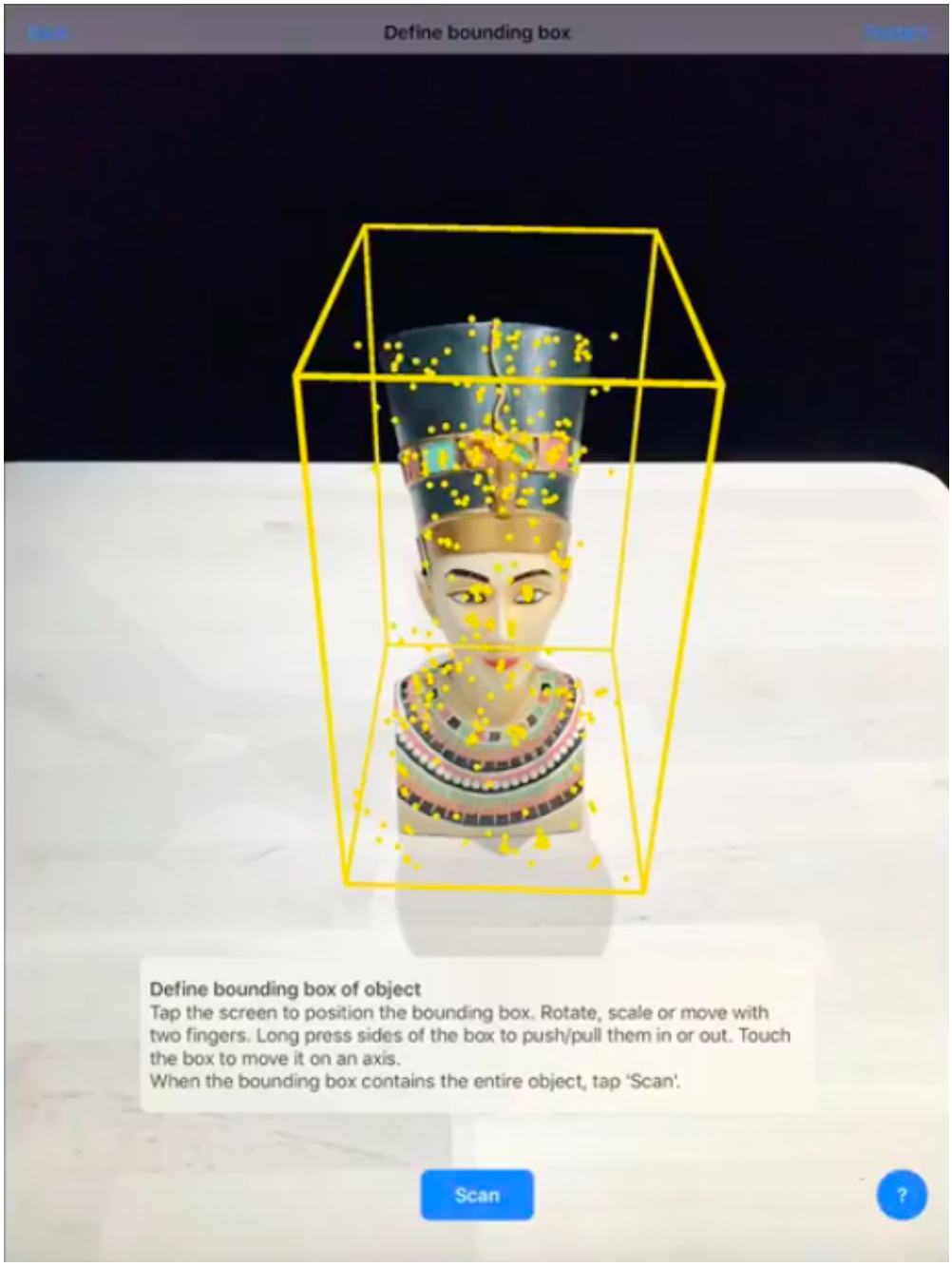

Рис. 2.14. Приклад сканування предмету

2.6.6. Розпізнавання та відстеження обличчя

З виходом iPhone X і його фронтальною камерою, Apple надала потужний інструмент для розпізнавання та відстеження облич. ARKit робить це, обчислюючи положення та орієнтацію обличчя для кожного кадру при 60 кадрах у секунду. Також було додано оцінку спрямованого світла. Пристрій використовує обличчя як світловий зонд, щоб отримати положення світла в сцені. Також, у ARKit додали Blend Shape, яка відстежує більше 50 конкретних функцій обличчя. Додано функцію виявлення погляду, яка відстежує праве та ліве око окремо, а також відстежується чи висунутий у користувача язик.

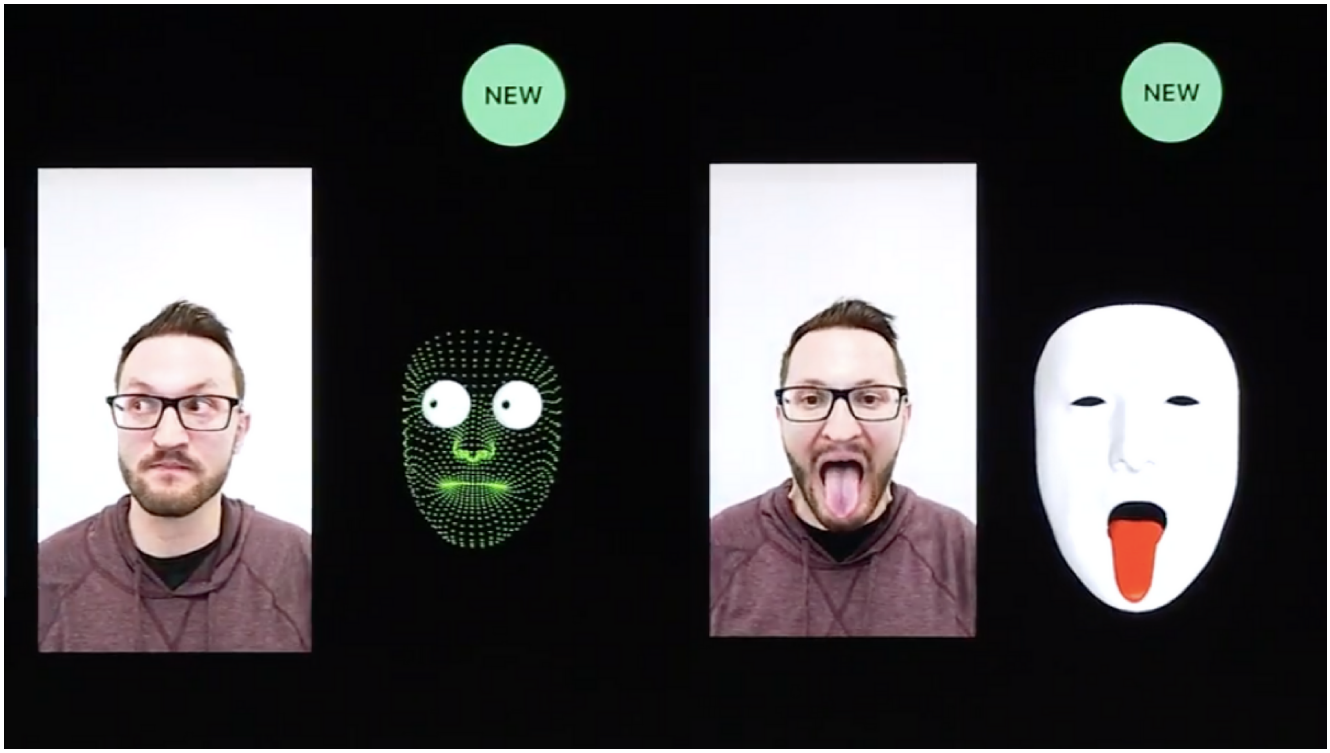

Рис. 2.15. Приклад розпізнавання та відстеження обличча, очей та язика

2.6.7. Спільний доступ

ARKit дозволяє створювати додатки, якими можуть користуватися декілька користувачів одночасно. Один з користувачів надає доступ до уже створеної сцени, яка зберігається у класі ARWorldMap. Все що залишається зробити іншому користувачеві, це завантажити надіслану сцену та знаходитися на місці сканування сцени першим користувачем. Простота ще й у тому що для з’єднання використовуються розповсюджені технології Bluetooth та Wi-Fi.

ВИСНОВКИ ДО РОЗДІЛУ 2

Виходячи з цього розділу, можна зробити висновок що найоптимальнішим алгоритмом для орієнтування пристрою у навколишньому середовищі є алгоритм SLAM. Він не має прив'язки до місця, як маркерна система, застосовується у сучасних бібліотеках для розробки додатків з технологією доповненої реальності, як наприклад ARKit та ARCore і має великий потенціал не тільки у сфері доповненої реальності.

Після проведення аналізу двох найпопулярніших бібліотек - ARKit та ARCore, було вирішено зупинитися на ARKit. По-перше, вона має кращу систему розпізнавання поверхонь та 3D-об'єктів. По-друге, ARKit працює на пристроях на базі операційної системи iOS, для якої і планувалася розробка програми, проте вона є нативною для цієї системи. На додаток потужність iPhone та iPad, та кількість підтримуваних девайсів - починаючи з iPhone 6S на базі процесора A9 2015 року випуску, та iPad Air 2 на базі процесора A8X 2014 року випуску, починаючи з версії iOS 11.0 з підтримкою ARKit 1.0, та iOS 12.0 з підтримкою ARKit 2.0, дає вагомий привід вибрати саме ARKit. Мовою програмування вибрано Swift, бо для iOS це нативна мова програмування.

РОЗДІЛ 3.

3. ВИКОРИСТАННЯ ПРОГРАМНИХ ЗАСОБІВ БІБЛІОТЕКИ ARKIT

Доповнена реальність дозволяє користувачам додавати 2D або 3D елементи до зображення з камери пристрою у реальному часі, й вони сприймаються як такі, що з’являються в реальному світі. ARKit поєднує в собі відстеження руху пристрою, захоплення сцени камери, розширену обробку сцени та зручність відображення для спрощення завдання побудов AR додатків. Ці технології можна використовувати для створення багатьох видів AR додатків за допомогою задньої або фронтальної камери пристрою на базі операційної системи iOS.

3.1. Доповнена реальність з використанням задньої камери

Найбільш поширені види додатків AR відображають вигляд з задньої камери пристрою iOS, доповненої іншим візуальним вмістом, що дає користувачеві новий спосіб побачити і взаємодіяти з навколишнім світом.

Клас ARWorldTrackingConfiguration надає таку можливість: ARKit відображає і відстежує реальний простір, який користувач захоплює камерою, і сканує його координатний простір для розміщення віртуального вмісту. Відстеження навколишнього середовища також пропонує функції, які роблять досвід AR більш реалістичним, наприклад, розпізнавання об'єктів та зображень у середовищі користувача та реагування на реальні умови освітлення.

3.2. Клас ARWorldTrackingConfiguration

ARWorldTrackingConfiguration - це конфігурація, що використовує задню камеру, відстежує орієнтацію та положення пристрою, виявляє поверхні реального світу та відомі зображення або об’єкти.

Всі конфігурації AR встановлюють відповідність між реальним світом пристрою і віртуальним 3D-координатним простором, де можна моделювати

```
class ARWorldTrackingConfiguration : ARConfiguration
```

Рис. 3.1. Оголошення класу ARWorldTrackingConfiguration

вміст. Коли додаток відображає цей вміст разом із зображенням у реальному часі, створюється ілюзія, що ваш віртуальний вміст є частиною реального світу.

Створення і підтримання цієї відповідності між просторами вимагає відстеження руху пристрою. Клас ARWorldTrackingConfiguration відстежує рух пристрою з шістьма ступенями свободи (6DOF): зокрема, три осі обертання і три осі руху (рух в x, y і z), як показано на рис. 3.2. Віртуальний об'єкт може залишатися на тому ж місці відносно реального світу, навіть коли користувач нахиляє пристрій, щоб бути вище або нижче об'єкта, або переміщує пристрій, щоб побачити сторони об'єкта.

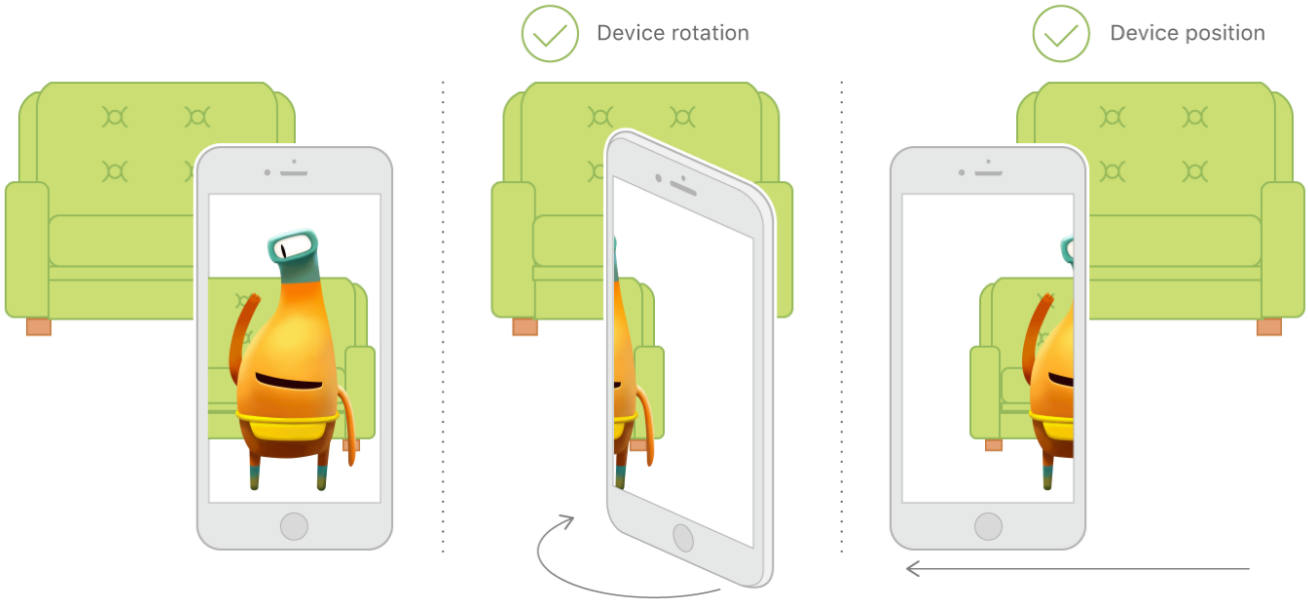

Рис 3.2. Приклад відстеження руху телефону

Сесії відстеження навколишнього середовища також дозволяють програмам розпізнавати чи взаємодіяти з елементами реальної сцени, видимими для камери:

1. Метод planeDetection використовується для того, щоб знайти реальні горизонтальні або вертикальні поверхні, додавши їх до сеансу як об'єкти ARPlaneAnchor.

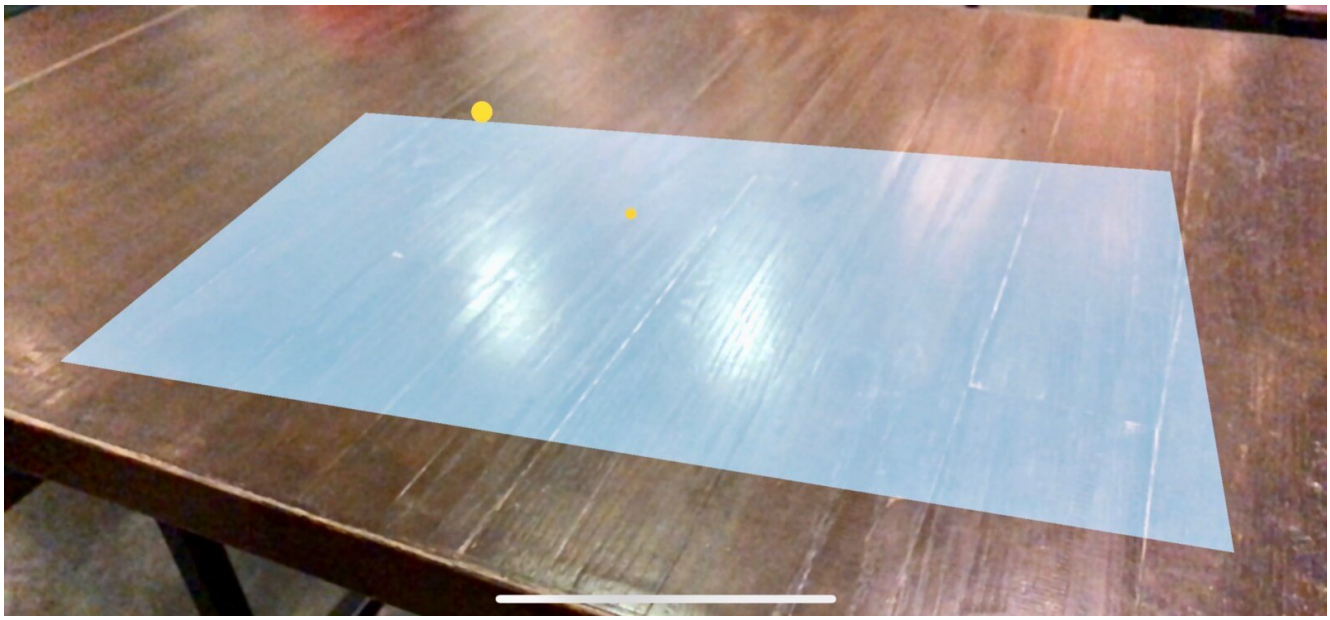

Рис. 3.3. Приклад виявлення поверхні

2. Метод detectionImages використовується для того, щоб розпізнавати і відстежувати рух відомих 2D-зображень, додаючи їх до сцени як об'єкти ARImageAnchor.

Рис. 3.4. Приклад розпізнавання 2D-зображення

3. Метод detectionObjects використовується для розпізнавання відомих 3D-об'єктів, додаючи їх до сцени як об'єкти ARObjectAnchor.

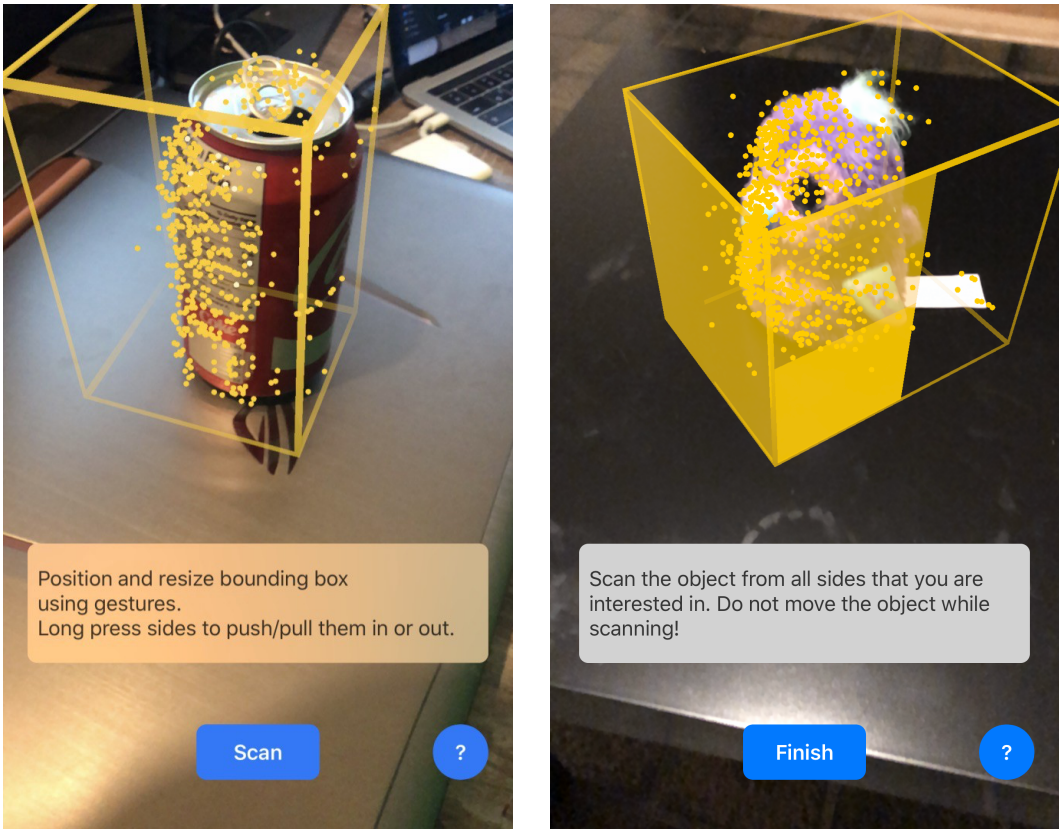

Рис. 3.5. Приклад розпізнавання та сканування 3D-об'єктів

4. Методи хіт-тестів використовуються на ARFrame, ARSCNView або ARSKView, щоб знайти 3D-позиції реальних точок, які корегуються з 2D-точкою у камері.

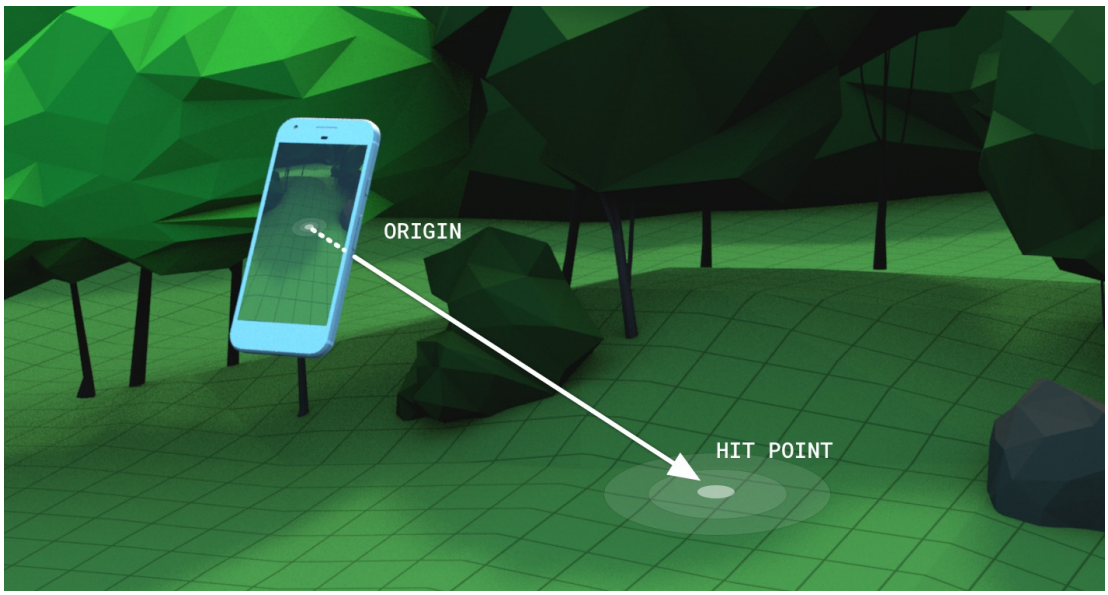

Рис. 3.6. Приклад роботи хіт-тесту у класі ARFrame

3.3. Доповнена реальність з використанням фронтальної камери

На iPhone X, Xs та Xs Max ARFaceTrackingConfiguration використовує фронтальну камеру TrueDepth для надання в реальному часі інформації про позицію і вираження обличчя користувача, яку можна використовувати у візуалізації віртуального вмісту. Наприклад, можна показати обличчя користувача з надітою реалістичною віртуальною маскою. Також можна використовувати дані виразу обличчя для анімації віртуальних символів, як це показано в програмі Animoji для iMessage.

Рис 3.7. Приклад роботи Animoji

3.4. Клас ARFaceTrackingConfiguration

ARFaceTrackingConfiguration - це конфігурація, яка відстежує рух і вираження обличчя користувача за допомогою камери TrueDepth.

```
class ARFaceTrackingConfiguration : ARConfiguration
```

Рис. 3.8. Оголошення класу ARFaceTrackingConfiguration

Конфігурація відстеження обличчя визначає обличчя користувача з огляду на фронтальну камеру пристрою. Під час виконання цієї конфігурації, сеанс AR виявляє обличчя користувача (якщо його видно на фронтальному зображенні камери) і додає до списку якорів об'єкт ARFaceAnchor, що представляє обличчя. Кожен якір на обличчі надає інформацію про положення та орієнтацію обличчя, його топологію та особливості, що описують міміку обличчя.

Рис 3.9. Приклад реалістичної віртуальної маски

Клас ARFaceTrackingConfiguration не надає жодних методів або властивостей, але підтримує всі властивості, успадковані від суперкласу ARConfiguration. Крім того, коли ви вмикаєте параметр isLightEstimationEnabled, конфігурація відстеження обличчя використовує виявлене обличчя як світловий зонд і надає оцінку спрямованого або природнього освітлення (об'єкт ARDirectionalLightEstimate), як зображено на рис 3.10.

Рис. 3.10. Приклад накладання реалістичного освітлення та тіней на віртуальну маску

3.5. Клас ARWorldMap

ARWorldMap відповідає за стан відображення простору і набір якорів з сесії AR, що відстежує світ.

Стан сеансу на карті світу включає в себе поінформованість ARKit про фізичний простір, у якому користувач переміщує пристрій (який ARKit використовує для визначення положення та орієнтації пристрою), а також будь-які об'єкти ARAnchor, додані до сеансу, які можуть представляти виявлені реальні ключові точки навколишнього середовища або віртуальний вміст, розміщений програмою.

Після того, як використати getCurrentWorldMap (completHandler :), щоб зберегти карту світу нинішньої сесії, ви можете присвоїти її властивості initialWorldMap конфігурації і використовувати run (_: options :), щоб почати іншу сесію з однаковою просторовою обізнаністю і якорями.

Зберігаючи карти світу та використовуючи їх для запуску нових сеансів, ваша програма може додавати нові можливості AR:

1. Досвід багатокористувацького AR. Можна створити загальну систему, відправивши архівовані об'єкти ARWorldMap до пристрою, розташованого поблизу користувача. За допомогою двох пристроїв, які відстежують одну карту світу, можна створити додатки для багатокористувацького використання,

де обидва користувачі можуть переглядати та взаємодіяти з однаковим віртуальним вмістом.

2. Неперервний досвід AR. Можна зберігати карту світу, коли програма стає неактивною, а потім відновлювати її наступного разу, коли програма запуститься в тому ж фізичному середовищі. Можна скористатися якорями від відновленої карти світу, щоб розмістити той самий віртуальний вміст на тих же позиціях із збереженого сеансу.

Рис. 3.11. Приклад багатокористувацької гри, за допомогою поширення об'єкта ARWorldMap

3.6. Вимірювання довжини у доповненій реальності

За допомогою класу ARWorldTrackingConfiguration та методу хіт-тестів програма ARExp будує віртуальну карту навколишнього середовища та знаходить поверхні, на яких користувач може розмістити точки, між якими знаходиться реальна відстань. За допомогою методу worldTransform, результати отримані після хіт-тестів, трансформуються у 3 значення - x, y та z. Це і є значення, потрібні програмі для того, щоб помістити точки на реальні поверхні у навколишньому світі.

На наступному скріншоті ми бачимо, як програма знайшла так звані якорі, помічені жовтим кольором, які позначають те, що знайдена поверхня, де можна розмістити точки для виміру відстані між ними, як показано на рис. 3.12.

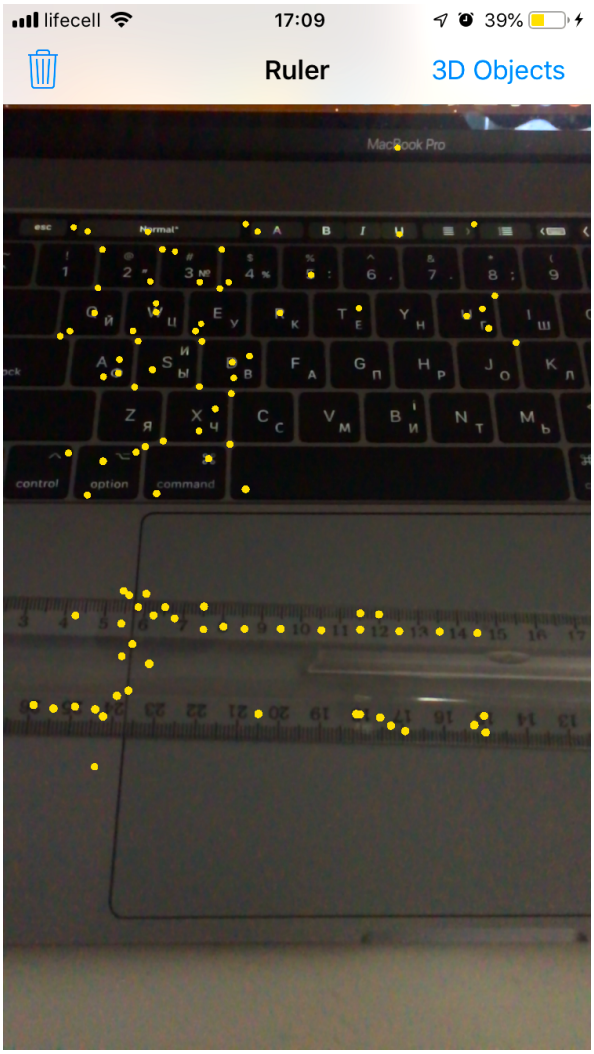

Рис. 3.12. Відображення якорів у програмі ARExp

Після натискання на екран, з’являється червона точка, яка розміщується на реальній поверхні у доповненій реальності для реалізму, точності та наочності вимірів реальних об’єктів. Після натискання у другий раз з’являється друга червона точка та відбувається розрахунок відстані між ними. Відстань відображається у метрах над другою поставленю точкою, як показано на рис. 3.13.

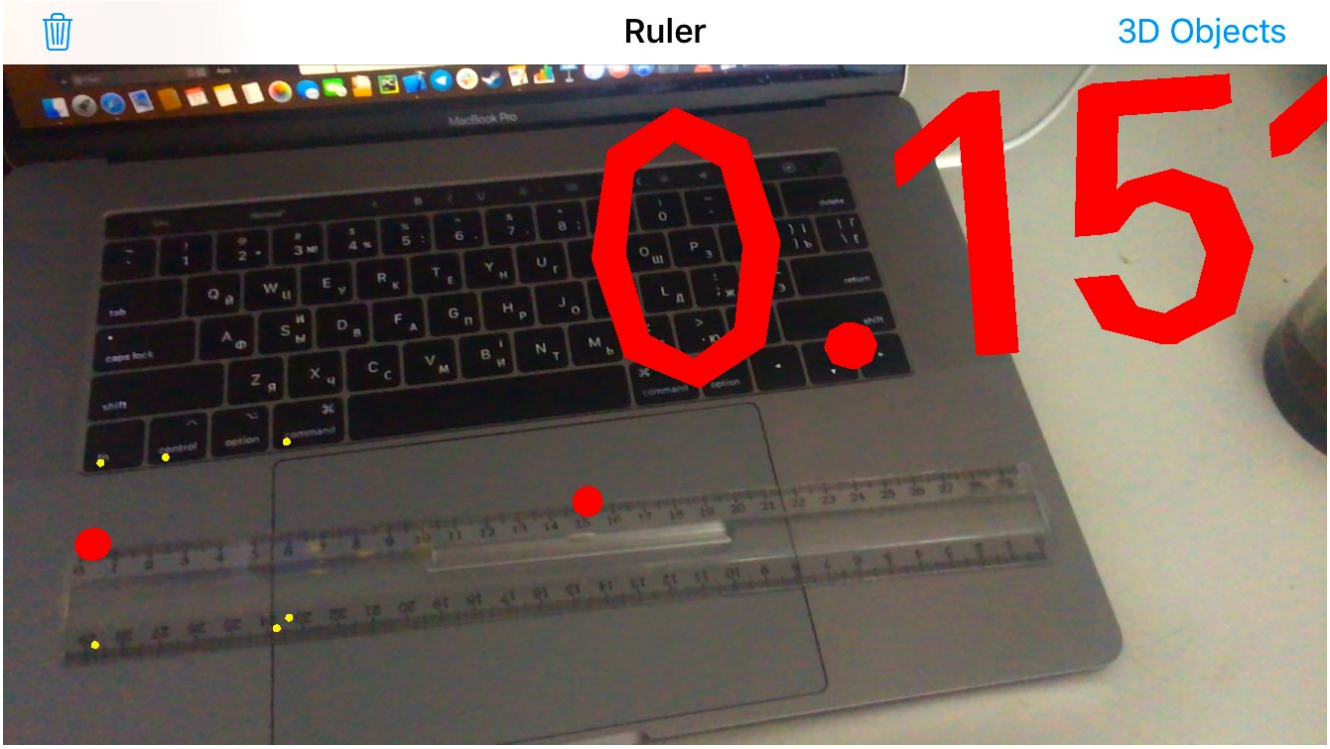

Рис. 3.13. Вимір довжини між двома точками у програмі ARExp

Як можна побачити з вищенаведеного скріншота, точність програми досить висока, що підтверджується лінійкою.

Також я провів тестування та порівняв свою програму, з програмою компанії Apple, та помітив що моя програма має змогу вимірювати відстань між точками на більшій віддаленості від камери, ніж аналог, як показано на рис. 3.14.

При натисканні у третій раз, попередні дві точки зникають та програма готова для виміру відстані між третьою та наступною точками і так далі, як зображено на рис. 3.15.

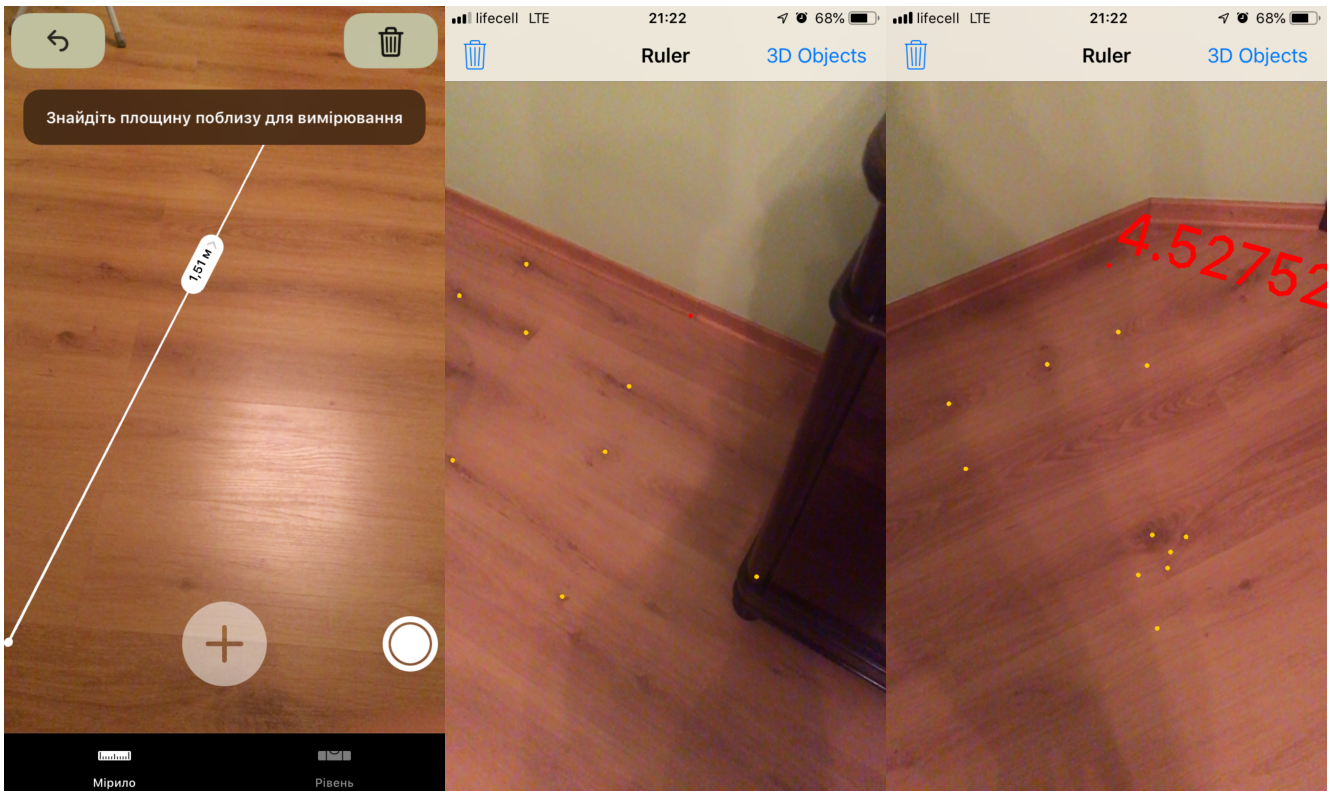

Рис. 3.14. Результат експерименту показує що аналог (зліва), може вимірювати довжину на меншій відстані від камери ніж ARExp (зправа)

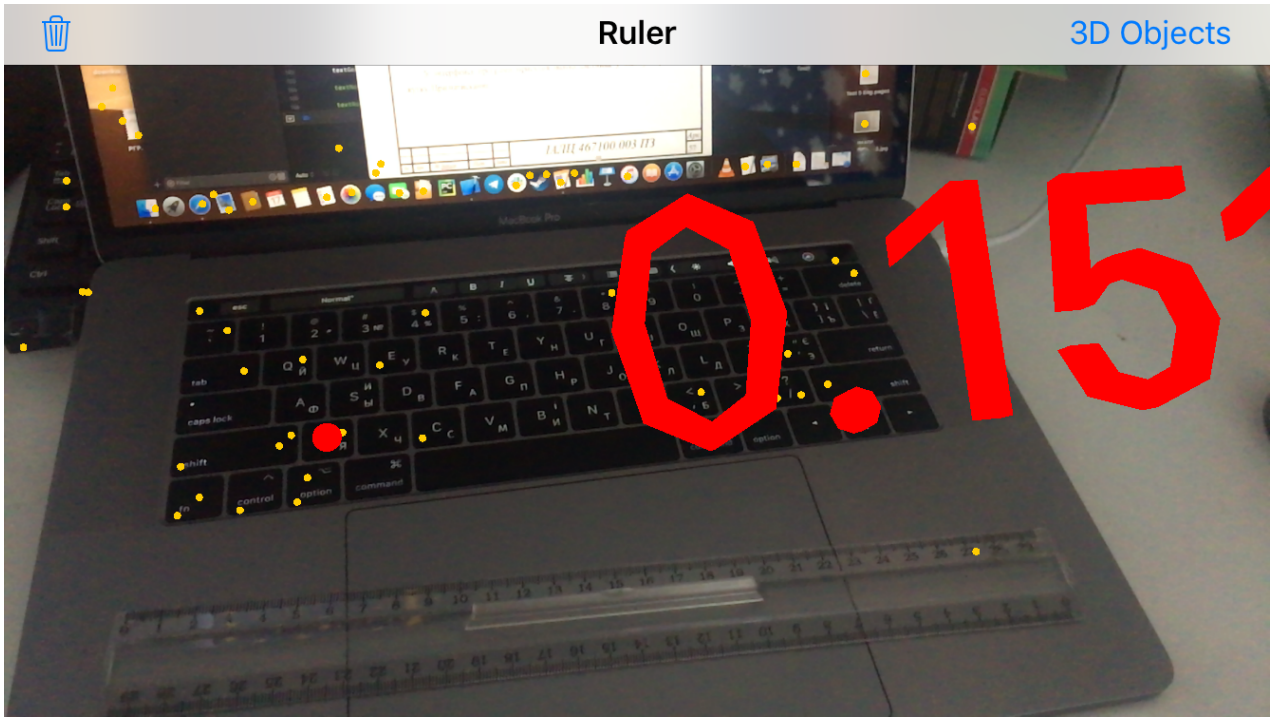

Рис. 3.15. Результат попереднього виміру залишається, а програма очікує наступні точки для виміру відстані між ними

У інтерфейсі програми присутня іконка смітника у лівому верхньому кутку. При натисканні на неї, усі точки та результат виміру зникають, як показано на рис. 3.16.

Рис. 3.16. Після натискання іконки смітника усі точки та результат зникають

Також у інтерфейсі присутня кнопка 3D Objects, яка розміщена у правому верхньому кутку. При натисканні на неї, програма переключається у режим пошуку відомих їй зображень, для відображення 3D моделей над ними.

3.7. Відображення 3D моделей над відомими зображеннями у доповненій реальності

За допомогою класу ARImageTrackingConfiguration програма ARExp відстежує навколишне середовище у пошуках відомих їй зображень, які зберігаються у ARReferenceImage. Після знаходження зображення, програма розміщує на нього поверхню, на яку зможе розмістити 3D модель, як зображено на рис. 3.17.

Рис. 3.17. Відображення поверхні над зображенням

Далі, на поверхню розміщується об’єкт, який прив’язаний до даного зображення, як показано на рис. 3.18.

Можна розміщувати декілька зображень у камеру і на всіх відомих програмі зображеннях будуть з’являтися 3D моделі, як показано на рис. 3.19.

У інтерфейсі присутнє відображення кількості кадрів на секунду у лівому нижньому кутку. Також у верхньому лівому кутку знаходиться кнопка, при натисканні на яку, програма повертається у режим виміру відстані між точками. Стани кожного з модулів зберігаються при перемиканні на інший, що робить користування програмою більш комфортним.

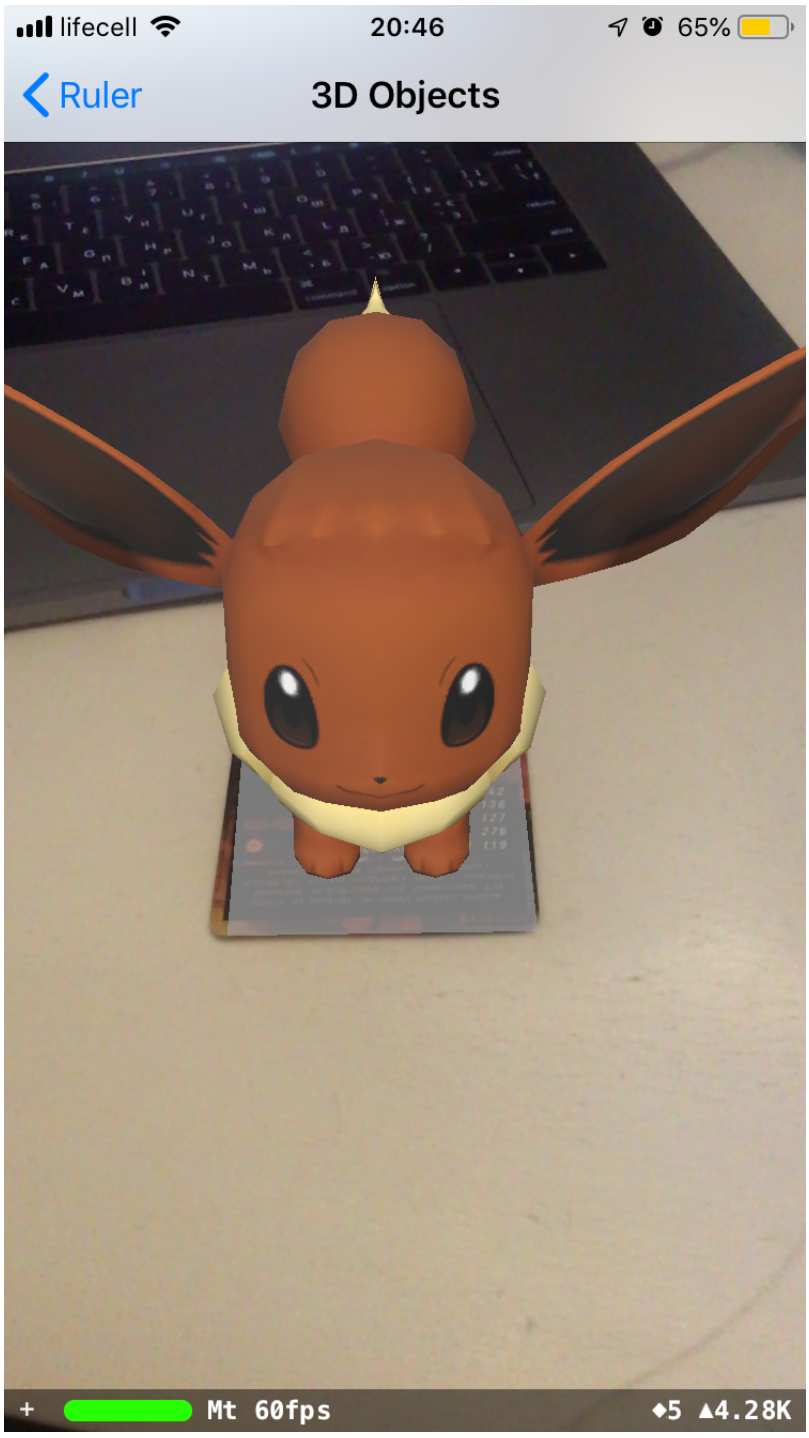

Рис. 3.18. Відображення 3D моделі над зображенням

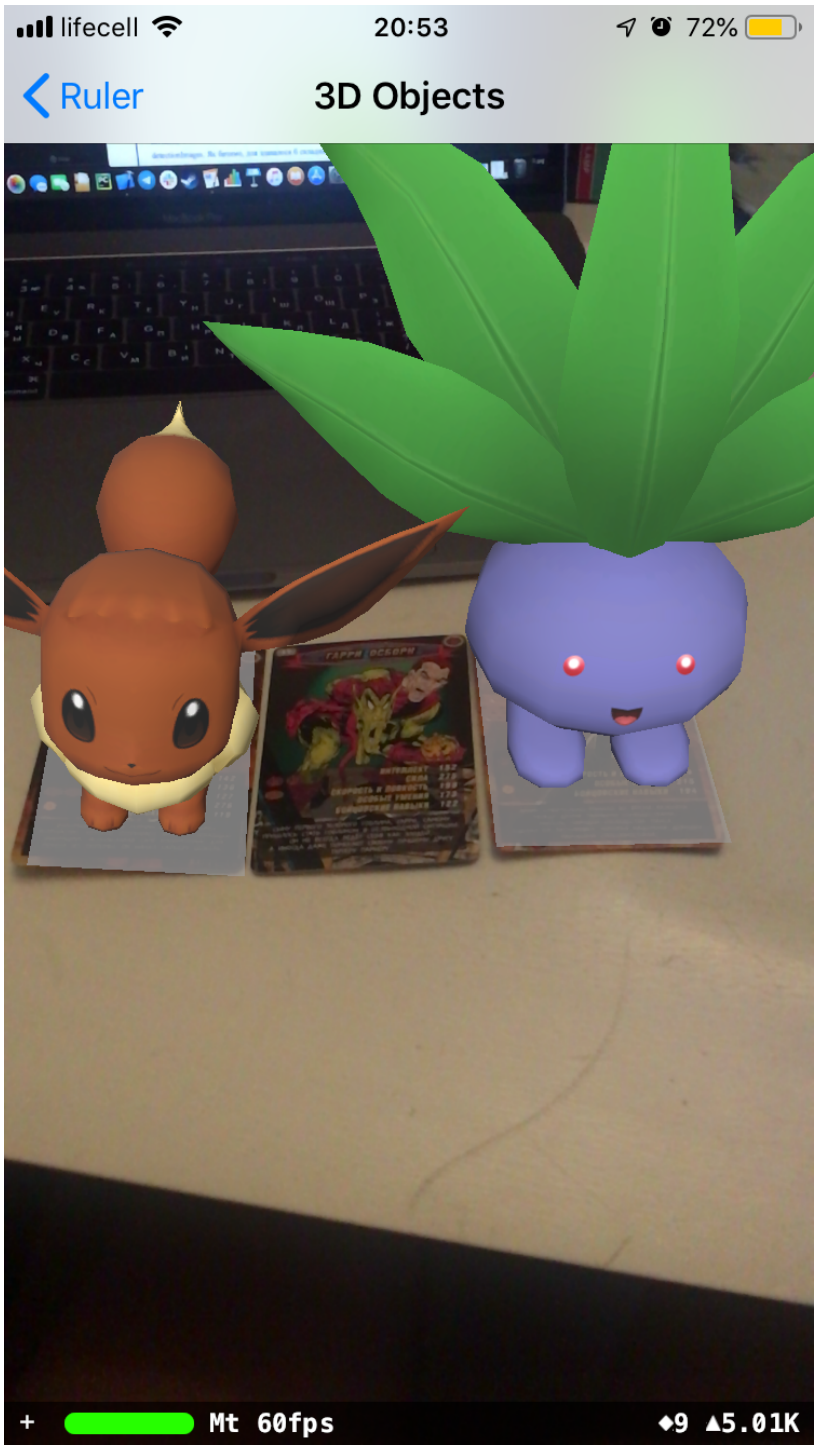

Рис. 3.19. Відображення 3D моделей лише над відомими зображенням

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі були розглянуті програмні засоби бібліотеки ARKit та описана робота програми ARExp. З усього різноманіття були вибрані ті інструменти, які потрібні для створення додатку, що вимірює довжини, а також для зображення 3D-об’єктів над 2D-мітками або зображеннями, а саме метод хіт-тестів для классу ARWorldTrackingConfiguration, так само як метод detectionImages. Як бачимо, для нетривіальних дій, не знадобилося багато методів та функцій. Це основа, яка дозволить програмі функціонувати у доповненій реальності. Ця бібліотека інтуїтивно зрозуміла та проста для розробника, і в той же час не втрачає у своїй потужності.

Під час тестування було виявлено, що аналог від компанії Apple має здатність вимірювати довжину між точками, які знаходяться на незначній відстані від камери, на відміну від реалізованої в цьому бакалаврському дипломному проекті програми, яка може робити теж саме, але тільки на більшій віддаленості від камери.

ВИСНОВОК

Результатом бакалаврської роботи є система, яка допомагає позбутися потрібності використовувати лінійку чи рулетку, та може без проблем продемонструвати 3D-об’єкти для їх легкого перегляду, без необхідності зайвих переміщень.

Розроблена система може бути корисна будівельникам, людям, які роблять ремонт, та просто як корисний інструмент або забава для дітей.

На даному етапі програма має великий потенціал для доопрацювання та збільшення функціоналу.

У подальших версіях планується додати функцію авто-розпізнавання країв об’єктів, що буде дуже корисно при вимірах предметів на кшталт плитки чи валіз. Також у планах застосувати технологію машинного навчання, та вивести продукт на якісно новий рівень. За допомогою цієї технології програма зможе розпізнавати різні об’єкти та виводити інформацію на головний дисплей у вигляді доповненої реальності.

Можна вважати, що програмний продукт зараз знаходиться на початковій стадії розробки і, при подальшій доробці, він може бути використаний у будівельному, розважальному, маркетинговому, інженерному та інших напрямках.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. ARKit Framework, Електронна документація [Електронний ресурс]. Режим доступу: https://developer.apple.com/documentation/arkit

2. ARCore Framework, Електронна документація [Електронний ресурс]. Режим доступу: https://developers.google.com/ar/develop

3. Mark Billinghurst, Adrian Clark and Gun Lee (2015), "A Survey of Augmented Reality", Foundations and Trends in Human–Computer Interaction [Електронний ресурс]: Vol. 8: No. 2-3, pp 73-272. Режим доступу: http://dx.doi.org/10.1561/1100000049

4. Ronald T. Azuma. A Survey of Augmented Reality // Presence: Teleoperators and Virtual Environments, 1997. P. 355–385.

5. Благовещенский И.А., Демьянков Н.А. Технології та алгоритми для створення доповненої реальності. Моделювання та аналіз інформаційних систем. [Електронний ресурс] 2013; 20(2):129-138. Режим доступу: https://doi.org/10.18255/1818-1015-2013-2-129-138

6. Shoaib, Huma & Jaffry, Syed Waqar. (2015). A Survey of Augmented Reality [Електронний ресурс]. Режим доступу: https://www.researchgate.net/publication/269464134_A_Survey_of_Augmented_Reality

7. Yeon Ma, Jung & Choi, Jong-Soo. (2007). The Virtuality and Reality of Augmented Reality. Journal of Multimedia. 2. 10.4304/jmm.2.1.32-37.

8. Höhl, Wolfgang & Broschart, Daniel. (2015). Augmented reality in architecture and urban planning [Електронний ресурс]. P. 111-117. Режим доступу: http://gispoint.de/fileadmin/user_upload/paper_gis_open/DLA_2015/537555011.pdf

9. Ruwan Egodagamage, Mihran Tuceryan, Distributed monocular visual SLAM as a basis for a collaborative augmented reality framework, Computers & Graphics, Volume 71, [Електронний ресурс] 2018, Pages 113-123, Режим доступу: https://doi.org/10.1016/j.cag.2018.01.002

10. Søren Riisgaard , Morten Rufus Blas. SLAM for Dummies A Tutorial Approach to Simultaneous Localization and Mapping [Електронний ресурс], Pages 6-29, Режим доступу: https://dspace.mit.edu/bitstream/handle/1721.1/119149/16-412j-spring-2005/contents/projects/1aslam_blas_repo.pdf

Додатки

ДОДАТОК 1

Система доповненої реальності для мобільної платформи iOS

Схема функціональна - блок-схема алгоритму

ІАЛЦ.467100.004 Д1

Аркушів 1

Київ 2019 р.

					ІАЛЦ.467100.004 Д1	
					Схема функціональна Блок-схема алгоритму	Літ. / Маса / Масштаб
Зм	Арк.	№ докум.	Підпис	Дата		Арк. / Аркушів
Розроб.		Філіпенко Р. О.				
Перевір.		Алещенко О.В.				
Н. контр.		Симоненко В. П.			Дипломна робота	КПІ ФІОТ кафедра ОТ гр. ІО-53
Затверд.						

ДОДАТОК 2

Система доповненої реальності для мобільної платформи iOS

Схема структурна - діаграма класів

ІАЛЦ.467100.005 Д2

Аркушів 1

Київ 2019 р.

					ІАЛЦ.467100.005 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Філіпенко Р. О.			Схема структурна Діаграма класів	Літ. / Маса / Масштаб
Перевір.		Алещенко О.В.				Арк. / Аркушів
Н. контр.		Симоненко В. П.			Дипломна робота	КПІ ФІОТ кафедра ОТ гр. ІО-53
Затверд.						

ДОДАТОК 3

Система доповненої реальності для мобільної платформи iOS

Схема взаємодії компонентів програми

ІАЛЦ.467100.006 ДЗ

Аркушів 1

Київ 2019 р.

					ІАЛЦ.467100.006 Д3		
Зм	Арк.	№ докум.	Підпис	Дата	*Схема взаємодії компонентів програми*	Літ.	Маса / Масштаб
Розроб.		Філіпенко Р. О.					
Перевір.		Алещенко О.В.				Арк.	Аркушів
Н. контр.		Симоненко В. П.			*Дипломна робота*	*КПІ ФІОТ кафедра ОТ гр. ІО-53*	
Затверд.							

ДОДАТОК 4

Система доповненої реальності для мобільної платформи iOS

Лістинг програми

ІАЛЦ.467100.007 Д4

Аркушів 4

Київ 2019 р.

```
//
//  ViewController.swift
//  ARExp
//
//  Created by Filipenko Roman on 12/04/19.
//  Copyright © 2019 Filipenko Roman. All rights reserved.
//

import UIKit
import SceneKit
import ARKit

class ViewController: UIViewController, ARSCNViewDelegate {

    @IBOutlet var sceneView: ARSCNView!

    var dotNodes = [SCNNode]()

    var textNode = SCNNode()

    override func viewDidLoad() {
        super.viewDidLoad()

        // Set the view's delegate
        sceneView.delegate = self

        sceneView.debugOptions = [ARSCNDebugOptions.showFeaturePoints]
    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)

        // Create a session configuration
        let configuration = ARWorldTrackingConfiguration()

        // Run the view's session
        sceneView.session.run(configuration)
    }

    override func viewWillDisappear(_ animated: Bool) {
        super.viewWillDisappear(animated)

        // Pause the view's session
        sceneView.session.pause()
    }

    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?)
{
        if dotNodes.count >= 2 {
            for dot in dotNodes {
                dot.removeFromParentNode()
            }

            dotNodes = [SCNNode]()
        }

        if let touch = touches.first {
            let touchLocation = touch.location(in: sceneView)
            let results = sceneView.hitTest(touchLocation,
types: .featurePoint)

            if let hitResult = results.first {
                addDot(at: hitResult)
            }
        }
    }
```

```
    func addDot(at hitResult: ARHitTestResult) {
        let dotGeometry = SCNSphere(radius: 0.005)
        let material = SCNMaterial()
        material.diffuse.contents = UIColor.red
        dotGeometry.materials = [material]
        let dotNode = SCNNode(geometry: dotGeometry)
        dotNode.position = SCNVector3(
            x: hitResult.worldTransform.columns.3.x,
            y: hitResult.worldTransform.columns.3.y,
            z: hitResult.worldTransform.columns.3.z
        )

        sceneView.scene.rootNode.addChildNode(dotNode)

        dotNodes.append(dotNode)
        if dotNodes.count >= 2 {
            calculate()
        }
    }

    func calculate() {
        let start = dotNodes[0]
        let end = dotNodes[1]

        let a = end.position.x - start.position.x
        let b = end.position.y - start.position.y
        let c = end.position.z - start.position.z

        let distance = sqrt(pow(a, 2) + pow(b, 2) + pow(c, 2))
        updateText(text: "\(abs(distance))", atPosition: end.position)
    }

    func updateText(text: String, atPosition position: SCNVector3) {
        textNode.removeFromParentNode()

        let textGeometry = SCNText(string: text, extrusionDepth: 1.0)
        textGeometry.firstMaterial?.diffuse.contents = UIColor.red

        textNode = SCNNode(geometry: textGeometry)

        textNode.position = SCNVector3(position.x, position.y + 0.01,
position.z)

        textNode.scale = SCNVector3(0.01, 0.01, 0.01)

        if let camera = sceneView.pointOfView {
            textNode.orientation = camera.orientation
        }

        sceneView.scene.rootNode.addChildNode(textNode)
    }

    @IBAction func clearAll(_ sender: UIBarButtonItem) {

        textNode.removeFromParentNode()

        for dot in dotNodes {
            dot.removeFromParentNode()
        }

    }

}
```

```
//
//  Objects.swift
//  ARExp
//
//  Created by Filipenko Roman on 12/04/19.
//  Copyright © 2019 Filipenko Roman. All rights reserved.
//

import UIKit
import SceneKit
import ARKit

class Objects: UIViewController, ARSCNViewDelegate {

    @IBOutlet var objectsView: ARSCNView!

    override func viewDidLoad() {
        super.viewDidLoad()

        // Set the view's delegate
        objectsView.delegate = self

        // Show statistics such as fps and timing information
        objectsView.showsStatistics = true

        objectsView.autoenablesDefaultLighting = true

    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)

        // Create a session configuration
        let configuration = ARImageTrackingConfiguration()

        if let imageToTrack = ARReferenceImage.referenceImages(inGroupNamed:
"Pokemon Cards", bundle: Bundle.main) {

            configuration.trackingImages = imageToTrack

            configuration.maximumNumberOfTrackedImages = 2

            print("Images Successfully Added")


        }

        // Run the view's session
        objectsView.session.run(configuration)
    }

    override func viewWillDisappear(_ animated: Bool) {
        super.viewWillDisappear(animated)

        // Pause the view's session
        objectsView.session.pause()
    }

    // MARK: - ARSCNViewDelegate

    func renderer(_ renderer: SCNSceneRenderer, nodeFor anchor: ARAnchor) ->
SCNNode? {

        let node = SCNNode()

        if let imageAnchor = anchor as? ARImageAnchor {
```

```
            let plane = SCNPlane(width:
imageAnchor.referenceImage.physicalSize.width, height:
imageAnchor.referenceImage.physicalSize.height)

            plane.firstMaterial?.diffuse.contents = UIColor(white: 1.0,
alpha: 0.5)

            let planeNode = SCNNode(geometry: plane)

            planeNode.eulerAngles.x = -.pi / 2

            node.addChildNode(planeNode)

            if imageAnchor.referenceImage.name == "eev-card" {
                if let pokeScene = SCNScene(named: "art.scnassets/eevee.scn")
{

                    if let pokeNode = pokeScene.rootNode.childNodes.first {

                        pokeNode.eulerAngles.x = .pi / 2

                        planeNode.addChildNode(pokeNode)
                    }
                }
            }

            if imageAnchor.referenceImage.name == "odd-card" {
              if let pokeScene = SCNScene(named: "art.scnassets/oddish.scn"){

                    if let pokeNode = pokeScene.rootNode.childNodes.first {

                        pokeNode.eulerAngles.x = .pi / 2

                        planeNode.addChildNode(pokeNode)
                    }
                }
            }

        }

        return node

    }

}
```